TIKOS

TIKI-DATA

T I KOS

1 6 B I T

O P E R A T I V S Y S T E M

for

T I K I - 1 0 0

****
* Tiki-Data A.S *
****

Tiki-Data A.S, Postboks 108 Refstad, N-0513 Oslo 5. Tlf: 02-154150

Forord

Denne manualen beskriver operativsystemet TIKOS for 16 bit tilleggskortet til Tiki-100 datamaskinen. TIKOS er laget som en integrert del av 8-bit operativsystemet TIKO for samme maskin. TIKO er beskrevet egen manual og forutsettes kjent av alle som skal benytte TIKOS.

TIKOS muliggjør kjøring av 16-bits programmer som er laget for CP/M-86 operativsystemet fra firmaet Digital Research i USA.

Manualen er organisert i følgende kapitler:

Kapittel 1 inneholder grunnleggende informasjon om TIKOS for terminalbrukere. Her beskrives bl.a. hvilke filer som må finnes på systemdisketten for å kunne kjøre 16-bit TIKOS programmer. Dette kapitlet bør leses grundig av alle som skal bruke TIKOS. Kapittel 2 gir en detaljert beskrivelse av TIKOS's programfunksjoner, og er beregnet for folk som skal utvikle nye programmer under TIKOS på maskinkode nivå.

TIKOS er et norsk system. Alle feilutskrifter er på norsk. TIKOS kan også fåes i engelsk versjon. Fordi alle TIKOS programmer startes fra (8-bit) TIKO, inneholder TIKOS <u>ingen</u> innebygde brukerkommandoer. I tillegg til å være en norsk versjon av CP/M-86 operativsystemet, inneholder TIKOS en rekke forbedringer og utvidelser i forhold til standard CP/M-86. Av disse kan kort nevnes:

- Automatisk disk "reset" ved diskett bytte.
- Største filstørrelse øket fra 8 Mb til 32 Mb.
- Største platestørrelse øket fra 8 Mb til 512 Mb.
- Utvidete rettefunksjoner ved linjeinntasting.
- Flere programfunksjoner.

Forslag til forbedringer og endringer mottas med takk, og kan sendes til

Tiki-Data A.S, Postboks 108 Refstad, N-0513 Oslo 5

NB! CP/M-86 er registrert varemerke til Digital Research.

**** Siste utskrift: 26/1-85 ****

Innholdsfortegnelse.

Innholdsfortegnelse, forts.

Side

Vedlegg:

Figurer:

Tabeller:

1. TIKOS brukerbeskrivelse.

1.1 Hva er TIKOS ?

TIKOS - TIKi OperativSystem - er operativsystemet for 16 bit tilleggskortet for Tiki-100 datamaskinen. Denne datamaskinen har en 8 bit Z80 som hovedprosessor. Tiki-100 er utviklet og produsert i Norge av firmaet Tiki-Data A.S.

TIKOS muliggjør kjøring av 16 bit programmer laget for det verdenskjente amerikansk-produserte CP/M-86 systemet fra Digital Research. TIKOS er funksjonsmessig kompatibelt med CP/M-86 versjon 2.2. I praksis betyr dette at ethvert program som kan kjøres på et standard CP/M-86 system også vil virke under TIKOS. TIKOS inneholder også en del forbedringer og utvidelser i forhold til CP/M-86.

TIKOS er integrert i 8-bit operativsystemet TIKO. Kommandotolkeren TKT i TIKO er utvidet til også å kunne lete etter 16 bit programmer ved programlasting. 16 bit TIKOS inneholder derved ingen spesielle brukerkommandoer. Lasting av 16-bit programmer foregår på samme måte som lasting av 8 bit programmer.

Eksempel:

```
a>>KAT *.TEK          (Laster 8-bit programmet KAT)
..
..
a>>XDIR *.TEK         (Laster 16-bit programmet XDIR)
```

Under selve kjøringen av 16-bit programmer, virker Z80 mikroprosessoren som en slave til 8088 prosessoren. Z80 utfører normalt alle inn/ut operasjoner mot periferutstyr (skjerm, skriver, etc.) etter kommando fra 8088.

1.2 TIKOS systemfiler.

TIKOS består av følgende filer som må finnes på systemplaten (vanligvis disk A) under kjøring:

1) 8088.COM - Z80 slaveprogram.

2) TIKOS.SYS - Inneholder selve 16 bit operativsystemet.

1.2.1 8088.COM filen.

8088.COM filen inneholder slaveprogrammet på Z80 under kjøring av 16 bit programmer. Programmet sørger også for å overføre alle nødvendige parametre til 16 bit programmet som skal kjøres. Videre vil 8088 programmet lese TIKOS.SYS filen inn til 8088 første gang et 16 bit program kjøres.

1.2.2 TIKOS.SYS filen.

TIKOS.SYS filen inneholder selve 16 bit operativsystemet som kjører på 8088 prosessoren. Filen lastes normalt kun en gang inn til 8088 første gang et 16 bit program skal kjøres.

Den viktigste delen av TIKOS.SYS filen er 16 bit filsystemet TIFS88 (=Tiki FilSystem for 8088). TIFS88 inneholder alle nødvendige programfunksjoner for å kunne kjøre CP/M-86 programmer.

1.3 Oppstart av TIKOS.

TIKOS startes opp automatisk av 8088 programmet første gang et 16 bit program skal kjøres. Når TIKOS startes, vil følgende melding komme på skjermen:

TIKOS (16 bit) - ver x.yz

(x.yz = versjonsnummer)

TIKOS kan også bli lest inn og startet på nytt ved følgende situasjoner:

- et program har "gått i frø" og ødelagt TIKOS
- etter at SHIFT BRYT er tastet

1.4 Program kommandoer.

Som nevnt tidligere er kommandotolkeren TKT i 8 bit TIKO utvidet til også å søke etter 16 bit programmer ved kommandoinntasting. 16 bit programfiler har etternavn CMD, mens 8 bit programmer har filtype COM. Brukere av TIKOS vil derved ikke merke forskjell på om et program er 8 eller 16 bit. Brukeren skriver bare programnavn etterfulgt av nødvendige programparametre.

Eksempel:

Anta at platen inneholder programfilene KAT.COM og XDIR.CMD som begge kan ha et flertydig filnavn som parameter. Programmet KAT.COM lastes ved å skrive

```
a>>KAT *.TIK
```

mens programmet XDIR.CMD startes ved å taste

```
a>>XDIR *.TIK
```

Hvis kommandotolkeren finner en fil på disken med etternavn CMD som passer med inntastet kommandonavn, blir slaveprogrammet 8088.COM lastet inn i Z80's hukommelse. Dette programmet sørger dernest for å gi besked til 16 bit TIKOS om at angitt 16 bit program skal kjøres. Selve programlastingen blir til slutt utført av TIKOS.

Kommandotolkeren KKT i 8 bit TIKO leter alltid <u>først</u> etter 8 bit programmer. Hvis et 8 bit og et 16 bit program har samme navn, vil følgelig 8 bit programmet bli kjørt. Den eneste måten å få lastet 16 bit programmet på er å skrive <u>8088 etterfulgt av programnavn.</u>

Eksempel:

Anta programfilene PROG.COM og PROG.CMD finnes på platen. Kommandoen

```
a>>PROG parameter
```

vil laste PROG.COM filen, mens kommandoen

```
a>>8088 PROG parameter
```

må skrives for å kjøre PROG.CMD programmet.

1.4.1 XDIR ffn

XDIR programmet skriver ut på skjermen en sortert filkatalog over alle filene som finnes på platen som passer med det flertydige filnavnet ffn. Utskriften vil også informere om størrelsen på hver fil, samt hvor mye dataplass som er ledig på platen. Programmet utfører stort sett det samme som 8 bit programmet KAT.

Følgende XDIR kommandoer er lovlige:

```
XDIR     *.*
XDIR                   (Samme som XDIR *.*)
XDIR     B:TIKI.*
A:XDIR   *.BA?
XDIR     ?.*           (Alle filer med fornavn på
                       kun EN bokstav)
XDIR     X.Y           (Hvor stor er filen X.Y?)
```

1.4.2 INTV

INTV programmet skriver ut innholdet av avbruddstabellen (="interrupt" vektor) til 8088 prosessoren. Programmet kan være nyttig under utvikling av nye programmer.

Eksempel:

```
a>>INTV
```

Vis 8088 avbrudds-tabell for TIKI-100 16 bit prosessor - ver 1.00

```
Avbrudd nr:   0  Verdi: 0040:27C9  TIKOS - Divisjon med 0
Avbrudd nr:   1  Verdi: 3BF6:09CA  ?     - Prosessor enkelt steg
Avbrudd nr:   3  Verdi: 3BF6:09C2  ?     - Innhopp til "debugger"
Avbrudd nr:   4  Verdi: 0040:27CE  TIKOS - Register overflyt
Avbrudd nr:  16  Verdi: 0040:27DF  TIKOS - IBM-PC spesial funksjon
Avbrudd nr: 160  Verdi: 0040:29AA  TIKOS - Les fra Z80 dataport
Avbrudd nr: 161  Verdi: 0040:29B2  TIKOS - Skriv til Z80 dataport
Avbrudd nr: 162  Verdi: 0040:29BA  TIKOS - Utfør Z80 subrutine
Avbrudd nr: 163  Verdi: 0040:29EF  TIKOS - Returner Z80 segment adresse
Avbrudd nr: 164  Verdi: 0040:29F3  TIKOS - Flytt 8-bit ord til/fra Z80 huk.
Avbrudd nr: 165  Verdi: 0040:29FC  TIKOS - Flytt 16-bit ord til/fra Z80 huk.
Avbrudd nr: 224  Verdi: 0040:0B06  TIKOS - Funksjons-innhopp adresse
Avbrudd nr: 225  Verdi: 0040:0B06  TIKOS - Funksjons-innhopp fra "debugger"
```

1.5 TIKOS feilmeldinger.

Feil under TIKOS kan enten oppstå under opplast eller under kjøring av 16 bit programmer.

1.5.1 Feil under opplast.

Følgende feilsituasjoner blir oppdaget under opplast av 16 bit programmer:

1) 8088.COM filen finnes ikke på systemplaten
2) TIKOS.SYS filen finnes ikke på systemplaten
3) 8088 tilleggskort mangler
4) Manglende 8088 hukommelse for programmet

1) 8088.COM filen finnes ikke på systemplaten.

Hvis brukeren har tastet en 16 bit program-kommando og filen 8088.COM ikke finnes på systemplaten, vil feilmeldingen

8088?

komme opp på skjermen. For å få kjørt 16 bit programmet, må brukeren kopiere inn 8088.COM filen til systemplaten.

2) TIKOS.SYS filen finnes ikke på systemplaten.

Hvis TIKOS.SYS filen ikke finnes på systemplaten ved første gangs kjøring av 16 bit programmer, vil feilmeldingen

Finner ikke TIKOS.SYS filen!

komme opp på skjermen. Brukeren må da kopiere TIKOS.SYS inn på systemplaten.

3) 8088 tilleggskort mangler.

TIKOS vil oppdage om 8088 prosessorkortet ikke er satt inn i maskinen, eller at kortet ikke virker. Feilmeldingen

8088 prosessor mangler!

vil bli skrevet ut.

4) Manglende 8088 hukommelse for programmet

TIKOS oppdager hvis programmet krever mer hukommelse enn del som finnes på 8088 kortet eller at programmet må lastes inn i et område som ikke er ledig. TIKOS vil skrive ut følgende feilmelding på skjermen:

Hukommelse mangler

1.5.2 Feil under kjøring.

Følgende type feilmeldinger kan komme på skjermen under kjøring av 16 bit programmer.

1) Filsystem feil.
2) Avbruddsfeil fra 16 bit prosessor.

1) Filsystem feil.

TIKOS kan oppdage 4 forskjellige feilsituasjoner under utførelse av filfunksjoner fra 16 bit programmer. Når slike feil oppstår, skriver TIKOS ut meldingen

Feil på <p>: <feilmelding>

på terminalen, hvor <p> er platelageret der feilen oppstod, A til P, og <feilmelding> er en av disse 4 meldingene:

Les/skriv
Gal plate
Fil KUN lesbar
KUN lesbar

"Les/skriv" feilmeldingen betyr at en fysisk feil under lesing eller skriving har oppstått på platen. Som oftest skyldes dette at disketten er ødelagt. Brukeren kommer ut av denne feil-situasjonen ved å taste KONTROLL-C eller en annen tast. KONTROLL-C stopper det kjørende programmet, mens programmet fortsetter hvis en annen tast slås. Brukeren kan imidlertid risikere at filkatalogen kan ødelegges hvis KONTROLL-C ikke tastes ved fysisk feil. Brukeren bør derfor være sikker på å ha sikkerhetskopi av disketten i slike tilfeller.

"Gal plate" feilmeldingen oppstår når 16 bit program prøver å logge seg inn på en plate som ikke finnes. Brukeren kvitterer med å trykke en tast. Dette fører til at programmet avbrytes.

"Fil KUN lesbar" feilen oppstår når et program prøver å skrive til en FIL som er skrivebeskyttet. Samme feil oppstår om filen blir forsøkt fjernet eller endret navn på. Brukeren kvitterer på feilmeldingen ved å trykke en tast, og programmet avbrytes.

"KUN lesbar" feilmeldingen skrives ut når et program prøver å skrive til en PLATE som er skrivebeskyttet. Brukeren kvitterer med å trykke en tast, og programmet avbrytes. Normalt vil en plate i TIKOS aldri automatisk bli skrivebeskyttet. Brukeren kan midlertidig skrivebeskytte en plate ved hjelp av 8 bit programmet "SETT".

2) Avbruddsfeil fra 16 bit prosessor.

TIKOS har innebygd mekanisme for å fange opp avbruddssituasjoner fra 8088 prosessoren. Slike avbrudd skyldes som regel en feil ved det kjørende programmet og fører til at programmet stoppes.

Når slike feil oppstår, skriver TIKO ut

**** <feil> i: ssss:oooo - Programmet stoppet ****

hvor <feil> er en av disse 3 meldingene:

Divisjon med 0
Register overflyt
IBM-PC interrupt 10 ulovlig
Ulovlig interrupt <nn>

"Divisjon med 0" feilmeldingen betyr at et program har prøvd å dividere med 0.

"Register overflyt" betyr at overflyt har oppstått i et av 8088 prosessorens regneregistre. For at dette skal skje, må det kjørende programmet ha gitt en spesiell "kommando" til 8088 (se 8088 manual). Denne feilsituasjonen oppstår sjelden.

"IBM-PC interrupt 10 ulovlig" feilmeldingen oppstår nå en på TIKI-100 prøver å kjøre programmer som er spesiallaget for IBM-PC. Feilsituasjonen oppstår nå programmet utfører 8088 instruksjonen "INT 10" (10 hex) for å utføre en IBM-PC spesialfunksjon.

"Ulovlig interrupt <nn>" feilmeldingen skyldes at det kjørende programmet har forsøkt å utføre en 8088 "INT nn" instruksjon, hvor nn er et hexadesimal tall.

Etter at en av de ovenfornevnte avbruddsfeil har oppstått, må TIKOS lastes inn på nytt for å kjøre flere 16 bit programmer.

2. TIKOS program beskrivelse.

Dette kapitlet omhandler TIKOS's interne organisering og alle TIKOS's programfunksjoner. Meningen med innholdet er å gi den nødvendige informasjon til å kunne utvikle nye programmer under TIKOS.

Innholdet forutsetter at leseren er kjent med 8088 maskinkode-programmering ("assembly"). Brukere av høynivåspråk, f.eks BASIC eller PASCAL, vil typisk utføre TIKOS funksjoner ved hjelp av spesielle språk-konstruksjoner i de enkelte høynivåspråk. Disse vil være beskrevet i brukerbeskrivelsene for språkene.

2.1 TIKOS moduler.

TIKOS er logisk inndelt i seks deler, Tiki Monitor Prom (TMP88), Tiki Inn-Ut Modul (TIUM88), TIki FilSystem (TIFS88), Tiki Program Laster (TPL88), Kommando og Program Område (KPO88), og Avbrudds-Tabellen (AT88).

TMP88 er det fast innbrente programmet på 8088-kortet som starter opp når strømmen slås på, den såkalte monitorprommen. Modulen inneholder kun et lite oppstart program for å kunne kommunisere med hovedprosessoren Z80.

TIUM88 inneholder funksjoner for å kople TIKOS's program-funksjoner til den elektronikk-nære programvaren i Z80 hukommelsen. Modulen inneholder bl.a en del disktabeller som kopieres fra Z80's TIUM modul før oppstart av alle 16 bit programmer.

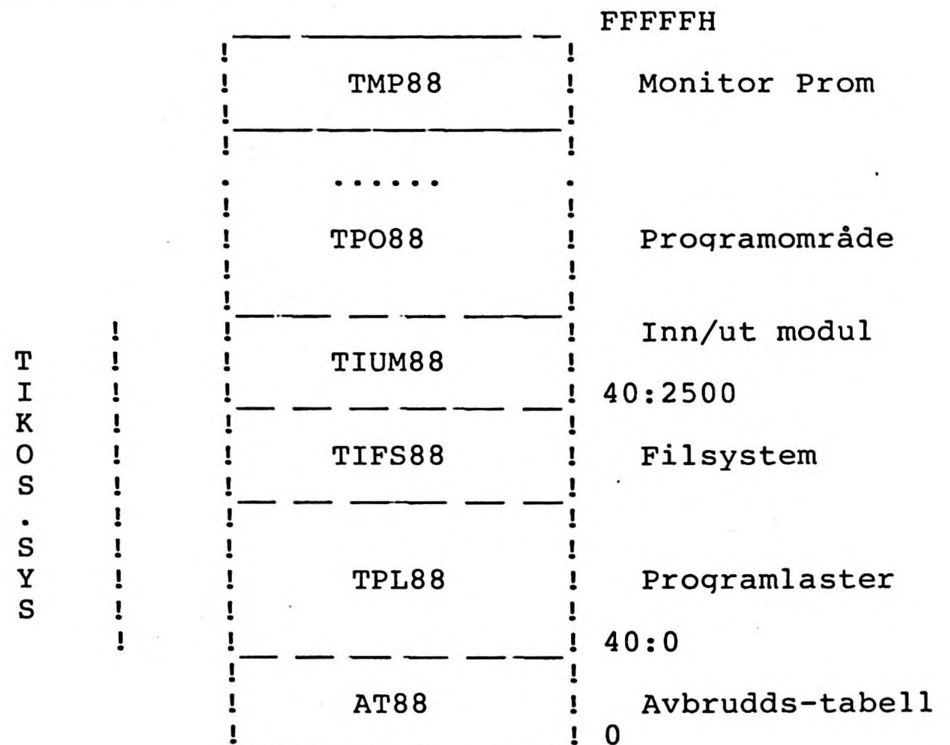
TIFS88 inneholder alle systemfunksjoner som kan utføres fra et program.

TPL88 inneholder funksjoner for å kunne kommunisere med Z80 prosessoren samt å kunne starte opplastingen av 16 bit programmer.

Kommando Programområdet KPO88 er den delen av hukommelsen der programkommandoer blir lastet inn og utført. Størrelsen av KPO88 kan variere noe fra forskjellige utgaver av Tiki maskinen.

Avbruddstabellen AT88 utgjør den nederste 1K av 8088 hukommelser og inneholder pekere til 8088's avbrudds-subrutiner.

Figuren nedenfor illustrerer hvordan TIKOS er plassert i hukommelsen:



Figur 2-1. 8088 hukommelseskart

TIKOS.SYS filen inneholder modulene TPL88, TIFS88 og TIUM88.

2.2 Program fil format

Alle 8088 programkommandoer er lagret på filer med etternavn "CMD". CMD filer består av et 128 tegns sektor hode etterfulgt av selve programmet. Sektorhodet består av 8 9-tegns gruppebeskrivelser (GB). Hver GB beskriver en del av det programmet som skal lastes. Figur 2-2 viser sektorhodet.

GB1	GB2	GB3	GB4	GB5	GB6	GB7	GB8	
0	9	18	27	36	45	54	63	.. 127

Figur 2-2. Sektor hode på programfiler

På figur 2-2 er GB1 til GB8 de enkelte gruppebeskrivelsene. Hvert gruppefelt beskriver en enhet av programmet som kan lastes separat, og har et format som vist i figur 2-3.

G-TYPE	G-LENGDE	A-BASE	G-MIN	G-MAX
0	1	3	5	7 / 8

Figur 2-3. Gruppebeskrivelse format

G-TYPE angir gruppe type, og er et 8-bit tall med verdi 1 til 8 som vist i tabell 2-1. Tabellen viser også de tilsvarende 8088 segment registrene for de enkelte gruppetypene. Segment registrene (untatt SS) vil bli satt opp til å peke på starten av de enkelte gruppene i hukommelsen når programmet starter.

Sektorhodet i en kommandofil kan ikke inneholde flere gruppebeskrivelser med samme G-TYPE verdi.

G-TYPE	Gruppe type	Segment register
01H	Kode gruppe	CS
02H	Data gruppe	DS
03H	Ekstra gruppe	ES
04H	Stack gruppe	SS
05H	Ekstra gruppe 1	--
06H	Ekstra gruppe 2	--
07H	Ekstra gruppe 3	--
08H	Ekstra gruppe 4	--

Tabell 2-1. Lovlige G-TYPE felt.

Alle resterende felt i gruppebeskrivelsen er 16-bit tall som angir antall 16-byte blokker (1 byte = 1 8-bit ord, 16 byte = 1 paragraf). Således beskriver gruppefeltene 20-bit verdier med de nederste 4 bit satt til 0. Tabell 2-2 beskriver 16-bit feltene i gruppebeskrivelsen.

Sektorhodet i 16-bit programfiler definerer den såkalte program-modellen programmet skal kjøres etter. Program-modeller er beskrevet i avsnitt 2.4. "Z80-modell" antas når sektorhodet bare inneholder en kode-gruppebeskrivelse. "Liten 8088 modell" antas når både kode- og datagruppe er tilstede, men ingen andre gruppebeskrivelser finnes. Mer enn to gruppebeskrivelser definerer såkalt "stor 8088 modell".

Felt	Beskrivelse
G-LENGDE	Angir antall paragrafer (16-byte blokker) i gruppen på selve programfilen. F.eks betyr en G-LENGDE på 80H at gruppen opptar 800H tegn på filen (2048 desimalt).
G-BASE	Definerer absolutt paragraf adresse i hukommelsen gruppen skal lastes til. Normalt vil G-BASE være 0. Operativsystemet vil da laste gruppen inn på et ledig område i hukommelsen.
G-MIN	Definerer minimum antall paragrafer som trengs til gruppen i hukommelsen. Hvis G-MIN er 0, antas G-MIN lik G-LENGDE.
G-MAX	Definerer maksimum antall paragrafer som gruppen kan bruke i hukommelsen. Hvis G-MAX er 0, antas G-MAX lik G-MIN.

Tabell 2-2. 16 bit felt i gruppebeskrivelsen.

Sektorhodet på programfiler blir automatisk generert av diverse 16 bit utviklings-programmer, f.eks. "GENCMD"-programmet fra Digital Research.

2.3 Program system område.

Når et 16 bit program lastes inn i hukommelsen, vil TIKOS sette opp en del parametre i det såkalte systemområdet til programmet. Systemområdet opptar de første 100 Hex (256 desimalt) 8-bit ordene av programmets data-gruppe (data-segment). Systemområdets innhold er vist i figur 2.4.

```
       0       1       2       3       4       5       6
       !-------+-------+-------+-------+-------+-------+
  0    !     KODE LENGDE       !  KODE BASE    ! MZ80  !
       !-----------------------------------------------!
  6    !     DATA LENGDE       !  DATA BASE    ! Ledig !
       !-----------------------------------------------!
 0C    !     EKSTRA LENGDE     !  EKSTRA BASE  ! Ledig !
       !-----------------------------------------------!
 12    !     STACK LENGDE      !  STACK BASE   ! Ledig !
       !-----------------------------------------------!
 18    !     EKSTRA1 LENGDE    !  EKSTRA1 BASE ! Ledig !
       !-----------------------------------------------!
 1E    !     EKSTRA2 LENGDE    !  EKSTRA2 BASE ! Ledig !
       !-----------------------------------------------!
 24    !     EKSTRA3 LENGDE    !  EKSTRA3 BASE ! Ledig !
       !-----------------------------------------------!
 2A    !     EKSTRA4 LENGDE    !  EKSTRA4 BASE ! Ledig !
       !-----------------------------------------------!
       !                                               !
       !                  Ikke brukt                   !
       !                                               !
 5C    !-----------------------------------------------!
       !                                               !
       !        STANDARD FILKONTROLL-BLOCK 1           !
       !                  (FKB1)                       !
 6C    !-----------------------------------------------!
       !                                               !
       !        STANDARD FILKONTROLL-BLOCK 2           !
       !                  (FKB2)                       !
 80    !-----------------------------------------------!
       !                                               !
       !                KOMMANDOHALE                   !
       !                                               !
       !          STANDARD INN/UT BUFFER               !
       !                                               !
       !------------------------------------------------
100
```

Figur 2.4. Program systemområde.

Feltene i program systemområde er definert på neste side:

- MZ80 feltet beskriver hvilken program-modell programmet har. Feltet kan ha følgende verdier:

  1 = Z80 modell
  0 = ikke Z80 modell (liten/stor 8088 modell)

- Lengde feltene angir lengden i antall 8-bit ord av de enkelte gruppene i programmet (kode,data, ekstra,stack,ekstral,..ekstra4). Lengde-feltene består av 3 8-bit ord der det første ordet utgjør de laveste sifrene i tallet. KODE-LENGDEN for Z80 modell programmer er aldri større enn 0FFFFH.

- Base feltene angir segment addressen til starten i hukommelsen av de respektive gruppene i programmet. Den fysiske 20 bit adressen framkommer ved å multiplisere segment adressen med 16.

- De to standard filkontroll-blokkene settes opp av TIKOS på de faste adressene 5CH og 6CH i systemområdet. Formatet på filkontrollblokker (FKB) er beskrevet i avsnitt 2.8. Når en kommandolinje på formen

  PROGRAM "filnavnl" "filnavn2"

  tastes, lages den første FKBen i adresse 5C Hex med filnavn feltene fylt inn med "filnavnl". Den andre FKBen blir laget i en del av den første FKBen (adresse 6C Hex), og inneholder "filnavn2". Denne FKBen må flyttes til en annen del av programhukommelsen før den kan brukes, mens den første kan brukes der den er. Uspesifiserte tegn i "filnavnl" og "filnavn2" blir fylt med blanke i de tilsvarende FKBene. Små bokstaver i filnavnene blir oversatt til store.

- Kommandohalen er en kopi av den delen av kommandolinjen som er etter selve programnavnet. Kommandohalen fylles inn i systemområdet fra adresse 81 Hex, med lengden på kommandolhalen fylt inn i hukommelses-cellen i adresse 80 Hex. Alle små bokstaver blir også her oversatt til store. Eksempelvis vil kommandolinjen

  a>>XDIR Z80.CPU Norsk

  resultere i følgende innhold fra adresse 80H:

```
0E 20 5A 38 30 2E 43 50 55 20 4E 4F 52 53 4B
       Z  8  0  .  C  P  U     N  O  R  S  K
```

  Kommandohale-området fra adresse 80 Hex brukes også som standard buffer for fil inn/ut operasjoner.

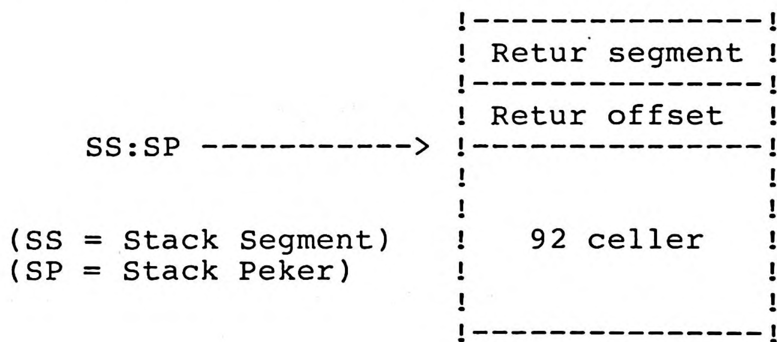
2.4 Program modeller.

Når et 16 bit program blir lastet inn i hukommelsen, vil segment-registrenes (CS,DS,ES,SS) startverdier bestemmes av hvilken modell programmet kjører under. Programmodellen er angitt i sektorhodet i CMD filen for programmet. Tre program modeller finnes, Z80 modell, liten 8088 modell, og stor 8088 modell. Tabell 2-3 sammenfatter de tre modellene.

Modell	Beskrivelse
Z80 modell	Kode- og datagruppe overlapper
Liten 8088 modell	Separat kode- og datagruppe
Stor 8088 modell	Mer enn to separate grupper

Tabell 2-3. Program modeller.

Under alle programmodellene vil TIKOS sette opp et internt stack-område på 96 hukommelses-celler som programmet kan benytte. Figur 2.5 viser det interne stack området i TIKOS.



Figur 2.5. Internt TIKOS stack-område

Første element i den interne stacken vil inneholde en returadresse til TIKOS slik at programmet kan returnere til TIKOS ved å utføre en "RETF"-("RETurn Far") instruksjon.

"Assembler" språk eksempel:

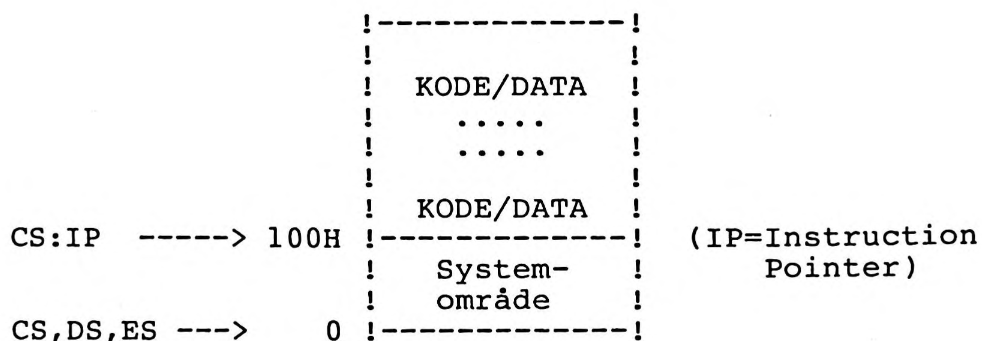
```
TIKOS     EQU     224          ; TIKOS interrupt
;
          CSEG                 ; Z80 PROGRAM MODELL
          ORG     100H
START:
          MOV     CL,1         ; LES ET TEGN FRA
          INT     TIKOS        ; TASTATURET

          ...                  ; (Mer programkode)

          RETF                 ; SLUTT PÅ PROGRAM
                               ; RETURNER TIL TIKOS
```

2.4.1 Z80 modell.

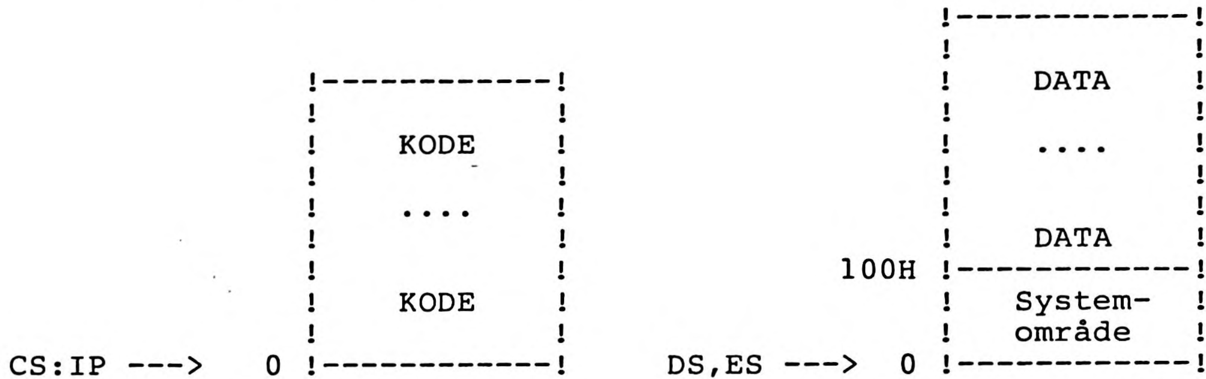
Z80 modellen brukes av programmer som er direkte oversatt fra 8 bit 8080 eller Z80 programmer. Z80 modellen består av kun en program gruppe som inneholder både kode, data og stack-områder. 8088's segment registre blir ved oppstart satt til å peke på starten av gruppen. Gruppens lengde er begrenset til 64K. Programeksemplet ovenfor viser et Z80 modell program. Figur 2.6 illustrerer modellen.



Figur 2.6. Z80 modell

2.4.2 Liten 8088 modell.

Programmer som kjører under denne modellen består av en uavhenging kode gruppe og data gruppe. Kode- og datagruppen er som regel, men behøver ikke være, begrenset i størrelse til 64K. Figur 2.7 illustrerer liten 8088 programmodell.



Figur 2.7. Liten 8088 modell.

Programeksemplet nedenfor viser et assembler program skrevet i liten 8088 modell:

```
TIKOS     EQU    224              ; TIKOS interrupt
;
          CSEG                    ; LITEN 8088 MODELL
          ORG    0
START:
          MOV    VAR,0            ; NULLSTILL VARIABEL

          MOV    DX,OFFSET FKB
          MOV    CL,15            ; ÅPNE FIL
          INT    TIKOS

          ...                     ; (Mer programkode)

          DSEG                    ; DEFINER DATAOMRÅDE

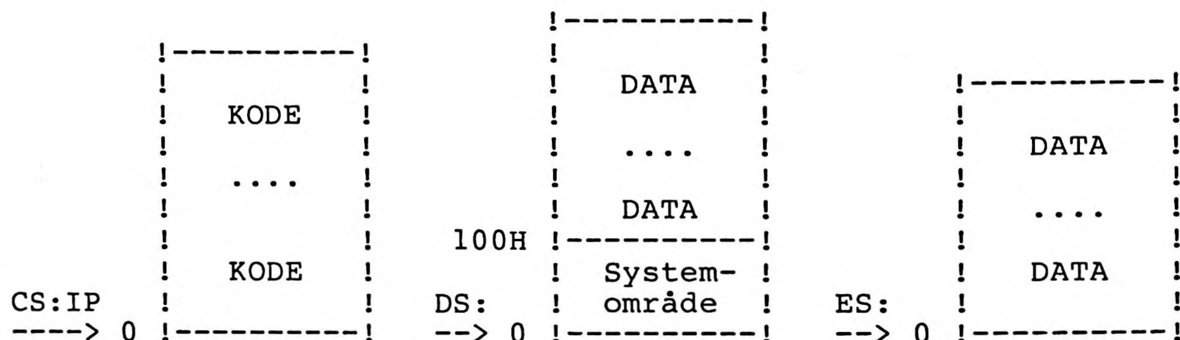
          ORG    5CH              ; STANDARD FKB
FKB       RB     36

          ORG    100H
VAR       DB     0                ; VARIABEL

END
```

2.4.3 Stor 8088 modell.

Programmer skrevet for denne programmodellen har en eller flere uavhengige programgrupper i tillegg til kode- og datagruppe. Som regel vil de ekstra gruppene være ekstra- og/eller stack-segment gruppe. Programmet vil i slike tilfeller som regel selv være i stand til å manipulere 8088's interne segment-registre. Figur 2.8 viser en typisk stor 8088 modell konfigurasjon etter programopplast.



Figur 2.8. Stor 8088 modell.

2.5 System-kall konvensjoner.

TIKOS systemfunksjoner tilgjengelig fra brukerprogrammer faller i fire klasser, enkle tegn inn-ut funksjoner, filsystem funksjoner, hukommelsesfunksjoner og TIUM-funksjoner.

Følgende type tegn inn-ut funksjoner finnes:

Les et tegn fra tastaturet
Test om nytt tegn er klart fra tastaturet
Les en linje fra tastaturet
Skriv et tegn på videoskjermen
Les et tegn fra port (serieport P2)
Skriv et tegn til port (serieport P2)
Skriv en sekvens av tegn på videoskjermen
Skriv et tegn på tilkoplet skriver

Følgende type filfunksjoner finnes:

- Initier filsystemet
- Velg ut et bestemt platelager
- Lag fil
- Åpne fil
- Lukk fil
- Søk i filkatalogen
- Fjern fil
- Endre filnavn
- Les sekvensielt eller randomisert fra fil
- Skriv sekvensielt eller randomisert til fil
- Finn ut hvilke platelagre som er logget inn
- Finn ut hvilken plate som er utvalgt
- Sett lese/skrive adressen i hukommelsen (DMA)
- Sett fil-attributt
- Sett filsystem feil modus
- Utfør et (annet) program
- Last inn program

Følgende type hukommelses funksjoner finnes:

- Bestem maksimal hukommelse tilgjengelig
- Tildel hukommelse
- Frigi hukommelse

TIUM88 basis inn/ut funksjoner:

Alle TIUM88 inn/ut funksjoner kan brukes fra brukerprogram. Normalt vil bare tegn-funksjoner brukes.

TIKOS programfunksjoner utføres ved hjelp av 8088 programinstruksjonen INT 224. Ved funksjonsinnhopp inneholder maskinregister CL alltid et funksjonsnummer. Parametre til funksjonen overføres i DX-registeret eller bare DL (DX=DH*256+DL). Parametre vil enten være et tall, f.eks et skjermtegn som skal skrives ut, eller en peker (adresse) til en hukommelseslokasjon som igjen kan inneholde flere parametre. For pekerparametre vil alltid segmentregisteret DS brukes som segment base.

TIKOS funksjonene vil returnere til brukerprogrammet med 8 bits returparametre i AL-registeret, og 16 bits returparametre i AX-registerparet. Dessuten vil alltid BX registeret være lik AX. I tillegg vil som regel segmentregister ES være satt til segmentet der selve TIKOS ligger (= 40 Hex).

Programeksempel:

```
          CSEG                    ; LITEN 8088 MODELL
          ORG     0
          JMP     START
;
OPSYS:    INT     224             ; TIKOS INNHOPP
          RET
;
START:
                                  ; DS:DX = FKB PEKER
          MOV     DX,OFFSET FKB
          MOV     CL,15           ; ÅPNE FIL
          CALL    OPSYS
          CMP     AL,0FFH         ; SJEKK OM FIL FUNNET
          JZ      NOTFOUND
                                  ; FIL FUNNET

          ...                     ; (Mer programkode)

NOTFOUND:                         ; FIL IKKE FUNNET
          MOV     CL,0            ; AVSLUTT PROGRAM
          MOV     DL,0
          JMP     OPSYS

          DSEG                    ; DEFINER DATAOMRÅDE

          ORG     5CH             ; STANDARD FKB
FKB       RB      36
          ORG     100H
                                  ; (Data område)
```

Tabell 2-4 viser alle TIKOS programfunksjoner med tilhørende funksjonsnumre.

0 - Last opp TIKO
1 - Tastatur inn
2 - Skjerm ut
3 - Port inn
4 - Port ut
5 - Skriver ut
6 - Direkte terminal inn/ut
7 - Hent INN/UT celle
8 - Sett INN/UT celle
9 - Skriv tegnsekvens
10 - Les tastatur linje
11 - Tastatur status
12 - Versjonsnummer
13 - Initier filsystemet
14 - Velgut plate
15 - Åpne fil
16 - Lukk fil
17 - Søk etter første
18 - Søk etter neste
19 - Fjern fil
20 - Les sekvensielt
21 - Skriv sekvensielt
22 - Lag fil
23 - Endre filnavn
24 - Hent innloggingstabell
25 - Hent utvalgt plate
26 - Sett lese/skrive inn/ut (offset) adresse
27 - Hent BLOKKTABELL adr.
28 - Skrivebeskytt plate
29 - Hent plater som er skrivebeskyttet
30 - Sett filattributt
31 - Hent platetabell adresse
32 - Sett/hent bruker nr.
33 - Les vilkårlig sektor
34 - Skriv vilkårlig sektor
35 - Beregn fil størrelse
36 - Beregn random sektor
37 - Logg ut plater
40 - Skriv vilkårlig sektor med NULL-stilling
45 - Sett filsystem feil-modus
47 - Start opp et annet program
50 - Utfør TIUM88 funksjon
51 - Sett lese/skrive inn/ut buffer segment
52 - Returner lese/skrive inn/ut adresse
53 - Angi maksimal tilgjengelig hukommelse
54 - Angi maksimal tilgjengelig hukommelse fra oppgitt base
55 - Alloker hukommelse
56 - Alloker hukommelse fra en bestemt base
57 - Frigi hukommelse
58 - Frigi all hukommelse
59 - Programlast

Tabell 2-4. TIKOS programfunksjoner.

2.6 Fil-type.

Etternavnet på en fil brukes ofte til å angi hva slags TYPE fil dette er, mens fornavnet brukes til å navngi de enkelte filene av denne filtypen. Tabellen nedenfor viser de fleste "standard" filtyper.

ASM - Assembly kildetekst
PRN - Listbar på skriver
HEX - Hexadesimal maskin kode
BAS - BASIC kildetekst
PAS - PASCAL kildetekst
COM - 8 bit programfiler
BAK - Kildetekst sikkerhetskopi
SYM - Symbol fil (brukt til programtesting)
BBB - BASIC programmer
C - C kildetekst
CMD - 16 bit programfiler
SUB - "BATCH" fil
A86 - 16 bit assembler kildetekst
H86 - 16 bit HEX-format filer
$$$ - Midlertidig fil
etc.

2.7 Fil-format.

Filer i TIKOS kan generelt ses på som en sekvens av opp til 262144 logiske TIKOS sektorer, hver på 128 8-bits tegn. Fildata vil alltid leses og skrives i faste blokker på en sektor. Sektorene er nummerert fra 0 til 262143, og tillater lagret over 32 millioner tegn pr. fil (et TIKOS platelager kan være på maksimalt ca. 530 millioner tegn). De logiske sektorene vil for TIKOS programmer se ut som om de er lagret sammenhengende på platelageret, men fysisk kan de være spredt på platelageret i usammenhengende områder. TIKOS's interne filorganisering er uinteressant for de aller fleste TIKOS programmer.

Filer vil normalt inneholde data lagret på enten tegn-format ("ASCII") eller binært-format. Eksempel på tegnfiler er kildetekstfiler, mens programmer alltid vil være lagret på binærfiler.

Tegnfiler er å betrakte som en lang sekvens av 7-bits ASCII-tegn organisert i linjer. Hver linje er en vilkårlig lang tegnsekvens etterfulgt av spesialtegnene CR ("Carriage Return" - tegnkode 13) og LF ("Line Feed" - tegnkode 10). Linjebredden vil normalt være fra 0 til ca. 130 tegn. En TIKOS sektor kan inneholde flere linjer, og en linje kan også strekke seg over flere (normalt maksimum 2) sektorer. Slutten på en tegnfil angis ved et slutt-på-fil spesialtegn (KONTROLL-Z - tallkode 26). Hvis siste tegn i filen faller på en sektorgrense, kan slutt-på-fil tegnet utelates.

Binærfiler består av et antall sektorer inneholdende binære 8 bits tall. Alle 8-bits tall, 0-255, kan lagres. Binære filer vil typisk være program- eller rådata-filer. Slutt-på-filen angis ved en sektorteller i filkatalogen og vil alltid falle på en sektorgrense. Programmer som leser binærfiler, må alltid sjekke på slutt-på-fil status fra TIKOS under lesingen.

2.8 Fil-kontrollblokk.

De fleste TIKOS filsystemfunksjonene trenger en peker til en filkontrollblokk (FKB) som funksjonsparameter i 8088's DX registerpar ved systeminnhopp. En FKB er en sammenhengende rekke av 36 8-bits tegn ("bytes") plassert i hukommelsem. Figur 2.9 viser FKB formatet.

```
+----------------------------------------------------------+
!pl!fl!f2! .. !f8!tl!t2!t3!sm!sl!s2!st!b0! ...!bn!ls!r0!rl!r2!
+----------------------------------------------------------+
 00 01 02      08 09 10 11 12 13 14 15 16      31 32 33 34 35
```

De enkelte felter i FKB'en er definert som følger:

pl - plate lager (0-16)

0 -> bruk innlogget plate for fil
1 -> bruk plate A for filoperasjon
2 -> bruk plate B for filoperasjon
. .
16 -> bruk plate P for filoperasjon

fl-f8 - fil fornavn på opptil 8 tegn. Hvert tegn er et 7-bits ASCII tegn (bits 0-6). Bit 7 i hvert tegn er 0. Kun store bokstaver bør brukes. Filnavnet avsluttes med et antall blanke tegn for filnavn med lengde mindre enn 8.

tl-t3 - fil etternavn (type) på opptil 3 tegn. Hvert tegn er et 7-bits ASCII tegn (bits 0-6). tl' og t2' er bit 7 av tl og t2, og er filattributter.

tl' = 1/0 -> Skrivebeskyttet/les og skriv
t2' = 1/0 -> Systemfil/Normal fil

sm - segment teller. Teller for TIKO's 16K segmenter. Normalt satt til 0 av brukerprogrammer, men kan variere fra 0 til 31 under filoperajoner.

sl - reservert for internt bruk.

s2 - reservert for internt bruk. Programmer bør sette s2 til 0 før fil-funksjonene ÅPNE,LUKK og SØK.

st - sektor teller innen hvert segment, 0-127.

b0-bl5 - Reservert for internt bruk.

ls - løpende sektor. Teller logisk sektor under sekvensielle filoperasjoner. Vanligvis satt til 0.

r0-r2 - logisk sektor nummer, 0 - 262143 Brukes kun under filoperasjoner som leser eller skriver vilkårlige sektorer. r0, rl or r2 utgjør et 18-bits sektornummer.

Figur 2.9 Fil-kontrollblokk (FKB).

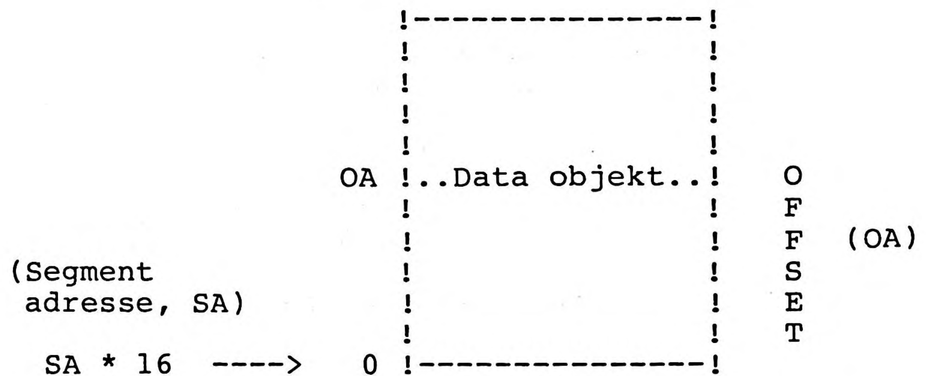
2.9 Programfunksjoner.

Denne delen beskriver alle TIKOS programfunksjoner samt funksjonsparametrene brukerprogrammer må overføre når de kaller de enkelte funksjonene. Funksjonsnumre og funksjonsparametre er angitt som hexadesimale tall. Hexadesimale tall skrives med bokstaven "H" bak tallet.

I beskrivelsen av de enkelte filfunksjonene er betegnelsene "offset adresse" og "segment adresse" mye brukt. "Offset adresse" (OA) er en 16 bit adresse relativt til starten av et hukommelses-segment. "Segment adresse" (SA) bestemmer startadressen til segmentet, og er et 16 bit tall som representer en fysisk 20 bit adresse med de nederste 4 bit satt til 0. "Virkelig" fysisk adresse (VA) framkommer således ved følgende formel:

VA = (SA*16) + OA

Figur 2.10 illustrerer sammenhengen mellom offset-, segment-, og virkelig adresse.



Fysisk adresse til data objekt: (SA*16) + OA

Figur 2-10. Segment/offset adresse.

```
**************************************************
*                                                *
*  Funksjon 0:     Avslutt program               *
*                                                *
**************************************************
*                                                *
*         Avslutter et TIKOS program.            *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 00H                         *
*      Register  DL: 0 eller 1                   *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Ingen                                     *
*                                                *
**************************************************
```

Denne funksjonen brukes til å avslutte TIKOS programmer. Hvis DL-registeret er satt til 1 ved innhopp, vil hukommelsen som okkuperes av programmet <u>ikke</u> bli frigitt. Ellers frigis hukommelsen som opptas av programmet.

```
*************************************************
*                                               *
*  Funksjon 1:     Tastatur inn                 *
*                                               *
*************************************************
*                                               *
*        Les et tegn fra tastaturet.            *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register  CL: 01H                        *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register  AL: Inntastet tegn (ASCII)     *
*                                               *
*************************************************
```

Tastatur-inn funksjonen leser neste tegn fra tastaturet til AL-registeret. Funksjonen venter til brukeren har tastet et tegn. Funksjonen gir også ekko av grafiske tegn ut på videoskjermen. Som grafiske tegn regnes bokstaver, tall, spesialtegnene, samt kontroll-tegnene CR, LF og Kontoll-H (flytter skrivemerket EN posisjon til venstre). Tabulator tegnet (Kontroll I) flytter skrivemerket til neste tabulator posisjon. Funksjonen tester også på spesial-tegnet kontroll-S som starter og stopper skjerm utskrift, samt kontroll-P som starter og stopper ekko til skriveren.

```
**************************************************
*                                                *
*  Funksjon 2:    Skjerm ut                      *
*                                                *
**************************************************
*                                                *
*         Skriv et tegn på videoskjermen         *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 02H                         *
*      Register  DL: Tegn (ASCII)                *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Ingen                                     *
*                                                *
**************************************************
```

Skjerm-ut funksjonen skriver tegnet i DL-registeret ut på videoskjermen. Tabulatortegn (kontroll-I) blir ekspandert til blanke, og funksjonen tester også på inntastet start/stopp skjerm tegn (kontroll-S) og skriver ekko tegn (kontroll-P).

```
************************************************
*                                              *
*  Funksjon 3:     Port inn                    *
*                                              *
************************************************
*                                              *
*          Les et tegn fra port (kanal P2)     *
*                                              *
************************************************
*                                              *
* Inngangsparametre:                           *
*                                              *
*       Register  CL: 03H                      *
*                                              *
* Utgangsverdier:                              *
*                                              *
*       Register  AL: Tegn fra port (ASCII)    *
*                                              *
************************************************
```

Port-inn leser neste tegn fra port P2 på Tiki datamaskinen til AL-registeret. Funksjonen venter til neste tegn er klar på porten. Denne funksjonen tilsvarer TIUM88 funksjonen PAPBÅNDI, som leser et tegn fra papirbånd. Papirbånd brukes imidlertid ikke lenger. Istedenfor leser funksjonen fra seriekanal P2.

```
***************************************************
*                                                 *
*  Funksjon 4:     Port ut                        *
*                                                 *
***************************************************
*                                                 *
*         Skriv et tegn til port (P2)             *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register  CL: 04H                          *
*      Register  DL: Tegn (ASCII)                 *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Ingen                                      *
*                                                 *
***************************************************
```

Skjerm-ut funksjonen skriver tegnet i DL-registeret ut på port P2 på Tiki datamaskinen. Denne funksjonen tilsvarer TIUM88 funksjonen PAPBÅNDU, som skriver et tegn til papirbånd. Papirbånd brukes imidlertid ikke lenger. Istedenfor skriver funksjonen til seriekanal P2.

```
***************************************************
*                                                 *
*  Funksjon 5:     Skriver ut                     *
*                                                 *
***************************************************
*                                                 *
*        Skriv et tegn på tilkoplet skriver       *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*       Register  CL: 05H                         *
*       Register  DL: Tegn (ASCII)                *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*       Ingen                                     *
*                                                 *
***************************************************
```

Skriver-ut funksjonen skriver tegnet i DL-registeret ut på tilkoplet skriver. Brukeren må selv sørge for at skriveren er slått på.

```
****************************************************
*                                                  *
*  Funksjon 6:     Direkte terminal inn/ut         *
*                                                  *
****************************************************
*                                                  *
*       Skriv et tegn på videoskjermen,            *
*       test om tegn er klart fra tastaturet,      *
*       les tegn fra tastaturet.                   *
*                                                  *
****************************************************
*                                                  *
* Inngangsparametre:                               *
*                                                  *
*      Register CL:  06H                           *
*      Register DL: 0FFH hvis tastatur status/inn  *
*      Register DL: 0FEH hvis tastatur status      *
*      Register DL: 0FDH hvis tastatur inn         *
*      Register DL: Tegn hvis skjerm ut            *
*                                                  *
* Utgangsverdier:                                  *
*                                                  *
*      Tastatur status/inn:                        *
*                                                  *
*        Register AL:                              *
*          = 0H hvis tastaturtegn ikke klart       *
*          = inntastet tegn hvis tegn klart        *
*                                                  *
*      Tastatur status:                            *
*                                                  *
*        Register AL:                              *
*          = 0H   hvis tastaturtegn ikke klart     *
*          = 0FFH hvis tastaturtegn klart          *
*                                                  *
*      Tastatur inn:                               *
*                                                  *
*        Register AL:                              *
*          = inntastet tegn                        *
*                                                  *
*      Skjerm ut:                                  *
*                                                  *
*        Ingen                                     *
*                                                  *
****************************************************
```

Direkte-terminal-inn/ut funksjonen brukes av programmer som ikke ønsker noen form for behandling av terminal inn/ut data. Ingen testing på spesialfunksjonene start/stopp skjerm (kontroll-S), ekko til skriver (kontroll-P) eller tabulator (kontroll-I) blir utført. Denne funksjonen er forøvrig noe utvidet i

forhold til standard CP/M-86.

Programmer kaller direkte-terminal-inn/ut funksjonen med å sette DL-registeret til en av fire verdier. Verdien i DL-registeret blir brukt til å bestemme hva slags inn/ut funksjon som skal utføres. Følgende verdier settes i DL-registeret ved innhopp:

0FFH - Funksjonen tester om brukeren har trykket en tast. Hvis ikke, returneres verdien 0H i AL-registeret. Hvis en tast er trykket, returneres tegnet i AL.

0FEH - Funksjonen tester bare om brukeren har trykket en tast. Hvis ikke, returneres 0H i AL-registeret. Hvis brukeren har slått en tast, returneres 0FFH i AL.

0FDH - Funksjonen leser neste tegn fra tastaturet. Funksjonen venter til et tegn er trykket. Tegnet returneres i AL-registeret.

Tegn - Hvis E-registeret verken er 0FFH, 0FEH eller 0FDH ved innhopp, tolkes DL-registeret som et tegn som skal skrives ut på videoskjermen. Ingen returparametre blir returnert i dette tilfellet.

```
***************************************************
*                                                 *
*  Funksjon 7:     Hent INN/UT celle              *
*                                                 *
***************************************************
*                                                 *
*     Hent TIKOS INN/UT celle fra Z80 hukommelse  *
*     Brukes til redirigering av terminal inn/ut  *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register  CL: 07H                          *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Register  AL: INN/UT celle                 *
*                                                 *
***************************************************
```

Hent-INN/UT-celle funksjonen henter fram aktuell verdi av systemets INN/UT celle som er lagret i Z80's hukommelse. Cellen blir brukt til å kunne dirigere terminal INN/UT enten til innebygget tastatur og tilkoplet videoskjerm eller til spesial-tilkoplet terminal over serielinje. Følgende bit i INN/UT cellen brukes:

Bit 0: 0 -> Tastatur INN tas fra innebygget tastatur.
1 -> Tastatur INN tas fra terminal tilkoplet over serielinje.

Bit 1: 0 -> Standard videoskjerm brukes for skjerm UT funksjoner.
1 -> Skjerm UT tegn sendes til tilkoplet terminal over serielinje.

Bit 2-7: Ikke brukt, men reservert.

```
**************************************************
*                                                *
*  Funksjon 8:     Sett INN/UT celle             *
*                                                *
**************************************************
*                                                *
*     Sett TIKOS INN/UT celle i Z80 hukommelsen. *
*     Brukes til redirigering av terminal inn/ut *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 08H                         *
*      Register  DL: INN/UT celle                *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Ingen                                     *
*                                                *
**************************************************
```

Sett-INN/UT-celle funksjonen setter ny verdi inn i systemets INN/UT celle lagret i Z80 hukommelsen. Cellen blir brukt til å kunne dirigere terminal INN/UT enten til innebygget tastatur og tilkoplet videoskjerm eller til spesialtilkoplet terminal over serielinje. Følgende bit i INN/UT cellen brukes:

Bit 0: 0 -> Tastatur INN tas fra innebygget tastatur.
1 -> Tastatur INN tas fra terminal tilkoplet over serielinje.

Bit 1: 0 -> Standard videoskjerm brukes for skjerm UT funksjoner.
1 -> Skjerm UT tegn sendes til tilkoplet terminal over serielinje.

Bit 2-7: Ikke brukt, men reservert.

```
**************************************************
*                                                *
*  Funksjon 9:    Skriv tegnsekvens              *
*                                                *
**************************************************
*                                                *
*       Skriv en sekvens av tegn på skjermen     *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 09H                          *
*      Register DX: Tegnsekvens offset adresse   *
*      Register DS: Tegnsekvens segment adresse  *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Ingen                                     *
*                                                *
**************************************************
```

Skriv-tegnsekvens funksjonen skriver en tegnsekvens lagret i hukommelsesadressen gitt i DX-registeret til skjermen. Et "$"-tegn brukes til å avslutte tegnsekvensen. Tabulatortegn blir ekspandert til blanke, og spesialfunskjonene start/stopp skjerm (kontroll-S tastet) og skriver ekko (kontroll-P tastet) blir utført.

```
*************************************************
*                                               *
*  Funksjon 10:   Les tastatur linje            *
*                                               *
*************************************************
*                                               *
*      Les en inntastet linje fra tastaturet    *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 0AH                         *
*      Register DX: Linjebuffer offset adresse  *
*      Register DS: Linjebuffer segment adresse *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Linjebuffer fylt med tegn                *
*                                               *
*************************************************
```

Les-tastatur-linje funksjonen brukes til å lese en linje fra tastaturet. Linjen avsluttes når enten linjebufferet er fullt eller når CR- eller LF tasten trykkes. Linjebufferet har følgende format:

```
DX:  +0 +1 +2 +3 +4 ......
    -------------------------------------
    !ml!at!tl!t2!t3!.......!tn!...!??!
    -------------------------------------
```

hvor "ml" er maksimal lengde på linjebufferen, "at" er antall inntastete tegn og "tl",.."tn" er de enkelte tegnene på linjen. Brukerprogrammer må alltid sette bufferlengden "ml" før funksjonen kalles. Hvis "at" er mindre enn "ml", vil resten av bufferet inneholde det samme som sto der ved funksjonsinnhopp.

Under inntastingen finnes de samme spesial/rettefunksjonene som brukeren har tilgjengelig under vanlig kommandoinntasting (se 8 bit TIKO manual).

Når funksjonen avsluttes med CR-tasten, vil skrivemerket på skjermen flytte seg til samme kolonneposisjon det hadde da funksjonen ble kalt (ikke nødvendigvis helt til venstre).

```
****************************************************
*                                                  *
*  Funksjon 11:    Tastatur status                 *
*                                                  *
****************************************************
*                                                  *
*   Test om nytt tegn er klart fra tastaturet      *
*                                                  *
****************************************************
*                                                  *
* Inngangsparametre:                               *
*                                                  *
*      Register  CL: 0BH                           *
*                                                  *
* Utgangsverdier:                                  *
*                                                  *
*      Register  AL: 0FFH hvis tegn er klart       *
*                      0H hvis tegn ikke er klart  *
*                                                  *
****************************************************
```

Tastatur-status funksjonen undersøker om brukeren har trykket en tast på tastaturet. Hvis et nytt tegn er klart, returneres 0FFH i AL-registeret. Hvis ikke, returneres 0 i AL. Funksjonen bør ikke brukes sammen med funksjon nr. 6 (direkte terminal inn/ut).

```
**************************************************
*                                                *
*  Funksjon 12:   Versjonsnummer                 *
*                                                *
**************************************************
*                                                *
*          Returner TIKOS versjonsnummer         *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*       Register  CL: 0CH                        *
*                                                *
* Utgangsverdier:                                *
*                                                *
*       Register AX: Versjonsnummer              *
*                                                *
**************************************************
```

Funksjon 12 returnerer versjonsnummer i henhold til standard CP/M-86. Versjonsnummer er 0022 Hex.

Funksjonen brukes av en del standard CP/M-86 programmer som også kan kjøres under bl.a MP/M-86 og CCP/M (flerbrukerutgaver av CP/M-86).

```
*************************************************
*                                               *
*  Funksjon 13:   Initier filsystemet           *
*                                               *
*************************************************
*                                               *
*         Start opp filsystemet på nytt         *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register  CL: 0DH                        *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register  AL: 0FFH hvis fil $*.* fil     *
*                        finnes på plate A      *
*      Register  AL: 0   ellers                 *
*                                               *
*************************************************
```

Funksjon 13 brukes av programmer til å sette filsystemet til en tilstand der alle platelagrene kan leses og skrives, bare plate A er logget inn og standard lese- og skriveadresse for filsystemet er 80 Hex i programmets systemområde. I TIKOS behøver funksjonen ikke brukes under bytting av disketter, fordi TIKOS har innebygd automatisk mekanisme for å finne ut om disketter har vært byttet.

Funksjonen returnerer 0FFH i AL-registeret hvis fil med fornavn som begynner med $ finnes på A platelageret. Ellers returneres 0 i AL.

```
**************************************************
*                                                *
*  Funksjon 14:    Velg ut plate                 *
*                                                *
**************************************************
*                                                *
*       Velg ut et bestemt TIKOS platelager      *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 0EH                         *
*      Register  DL: Platelager, 0-15            *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Ingen                                     *
*                                                *
**************************************************
```

Velgut-plate funksjonen logger inn platelageret gitt av DL-registeret som STÅENDE plate. DL=0 tilsvarer plate A, DL=1 angir plate B, helt opp til DL=15 som velger ut plate P. Den utvalgte platen vil i etterfølgende filsystem-kall aksesseres av filfunksjoner der "pl" feltet i filkontrollblokken (FKB) er 0. "pl" verdier mellom 1 og 16 vil imidlertid ignorere den stående platen og direkte aksessere plate A til P.

Hvis disketter byttes mens de er logget inn, vil TIKOS automatisk oppdage dette, og logge inn platen på nytt.

```
***************************************************
*                                                 *
*  Funksjon 15:   Åpne fil                        *
*                                                 *
***************************************************
*                                                 *
*   Åpne en fil for etterfølgende filfunksjoner   *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 0FH                           *
*      Register DE: FKB offset adresse            *
*      Register DS: FKB segment adresse           *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Register AL: 0-3  hvis filen ble åpnet OK  *
*                   0FFH hvis filen ikke finnes   *
*                                                 *
***************************************************
```

Åpne-fil funksjonen forbereder en fil for etterfølgende les- og skrivoperasjoner. Filsystemet søker etter filen i katalogen under aktiv bruker på det angitte platelageret. Feltene i posisjon 1 til 14 i filkontrollblokken (FKB) må stemme overens med filen i katalogen for at filen skal finnes. Et spørsmålstegn i en feltposisjon i FKBen samsvarer med ethvert tegn på tilsvarende posisjon i katalogen. Vanligvis vil programmer kun spesifisere entydige filnavn i et åpne-fil kall. Likeledes vil "sm" og "s2" feltene (se avsnitt 2.5) vanligvis settes til 0 før funksjon 15 kalles.

Hvis filen finnes på platen, blir kataloginformasjon kopiert til feltene "b0" til "bn" i FKBen, slik at FKBen kan brukes i etterfølgende lese og skrive operasjoner (Merk: Ingen filer kan brukes før de er åpnet).

Funksjon 15 returnerer med AL-registeret satt til verdi 0 til 3 hvis filen ble korrekt åpnet. Hvis filen ikke finnes returneres 0FFH til AL.

Hvis FKBen inneholder spørsmålstegn, blir den første filen i katalogen som passer åpnet.

Programmer som skal lese og skrive sekvensielt fra den første logiske filsektoren, må sette sektor-telleren "ls" i FKBen til 0 etter at funksjon 15 er brukt.

```
*****************************************************
*                                                   *
*  Funksjon 16:    Lukk fil                         *
*                                                   *
*****************************************************
*                                                   *
*    Lukk en fil etter utførte filoperasjoner       *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*       Register CL: 10H                            *
*       Register DX: FKB offset adresse             *
*       Register DS: FKB segment adresse            *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*       Register AL: 0-3  hvis filen ble lukket OK  *
*                    0FFH hvis filen ikke finnes    *
*                                                   *
*****************************************************
```

Lukk-fil funksjonen utfører det motsatte av en åpne-fil funksjon. Hvis DX-registeret adresserer en FKB som tidligere ble brukt til å åpne eller lage en fil (se funksjon 15 og 22), vil lukk-fil funksjonen sørge for å kopiere all nødvendig fil-informasjon fra FKBen til filkatalogen.

Lukk-fil funksjonen bruker samme søkemekanisme i fil-katalogen som åpne-fil funksjonen.

Funksjon 16 returnerer med AL-registeret satt til verdien 0 til 3 hvis filen finnes i katalogen og har blitt tidligere åpnet. Hvis filen ikke finnes i katalogen, returneres 0FFH i AL.

Filer som bare leses behøver ikke lukkes etter at de har vært brukt (det er imidlertid god skikk å gjøre det). Filer som har vært skrevet må imidlertid alltid lukkes etter bruk for at filkatalogen skal oppdateres med nødvendig informasjon. Hvis ikke, vil det som er skrevet mistes.

```
**************************************************
*                                                *
*  Funksjon 17:    Søk etter første              *
*                                                *
**************************************************
*                                                *
*    Søk etter første forekomst av en fil        *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*       Register CL: 11H                         *
*       Register DE: FKB offset adresse          *
*       Register DS: FKB segment adresse         *
*                                                *
* Utgangsverdier:                                *
*                                                *
*       Register AL: 0-3  hvis en fil funnet     *
*                    0FFH hvis ingen fil finnes  *
*                                                *
**************************************************
```

Søk-etter-første funksjonen søker etter første forekomst av en fil i katalogen på en plate under aktivt brukernummer. Verdien 0FFH (255 desimalt) returneres hvis ingen fil finnes. Ellers vil AL-registeret inneholde KATALOG-KODEN 0,1,2 eller 3. Hvis en fil samsvarende med FKBen finnes, vil hukommelsen gitt ved filsystemets løpende lese/skrive buffer inneholde den TIKOS filkatalog sektoren (128 tegn) som inneholder den søkte filen. Hvert filelement tar 32 tegn i katalogsektoren, og samsvarer med elementene i FKBen. Adressen til det søkte katalogelementet vil kunne beregnes ut fra følgende formel:

adresse = lese/skrive buffer + 32*katalog-kode

Brukerprogrammer kan undersøke filkatalogelementene særskilt, for derved å kunne bestemme f.eks filnavn, filtype og størrelse.

Spørsmålstegn i posisjon 1 (filnavn tegn nr. 1) til og med 12 (segment teller) i FKBen passer med et hvilket som helst tegn på samme sted i katalogelementet på innlogget plate eller særskilt spesifisert plate. Hvis "pl" (plate lager) feltet i FKBen også inneholder et spørsmålstegn, vil ethvert katalogelement, ledig eller opptatt, tilhørende et hvilket som helst brukernummer, på innlogget plate passe. Funksjon 17 vil derved kunne brukes til å starte gjennomlesing av fysisk alle katalogelementene. Hvis "pl" feltet ikke inneholder spørsmålstegn, vil "s2" feltet alltid bli satt til 0.

```
**************************************************
*                                                *
*  Funksjon 18:   Søk etter neste                *
*                                                *
**************************************************
*                                                *
*    Søk etter neste forekomst av en fil         *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 12H                          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL: 0-3  hvis en fil funnet      *
*                  0FFH hvis ingen fil finnes    *
*                                                *
**************************************************
```

Søk-etter-neste funksjonen er lik funksjon 17 bortsett fra at gjennomsøkingen i filkatalogen starter fra det siste elementet som passet med spesifisert FKB i søk-etter-første funksjonen. Funksjon 18 returnerer katalog kode i AL-registeret som i funksjon 17.

Funksjon 18 må etterfølge enten funksjon 17 eller et annet funksjon 18 kall. Ingen andre filsystemkall, unntatt skjerm- og skriver-kall, må komme imellom to søkekall.

```
**************************************************
*                                                *
*  Funksjon 19:   Fjern fil                      *
*                                                *
**************************************************
*                                                *
*             Fjern en fil fra platen            *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 13H                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL: 0-3  hvis fil(er) fjernet    *
*                  0FFH hvis ingen fil finnes    *
*                                                *
**************************************************
```

Fjern-fil funksjonen fjerner filer som passer med spesifisert FKB. Angitt filnavn kan være flertydig for å kunne fjerne flere filer på en gang. Platelager feltet ("pl") i FKBen kan ikke inneholde spørsmålstegn.

Funksjon 19 returnerer med 0FFH (255 desimalt) i AL-registeret hvis ingen fil som samsvarer med FKBen finnes i katalogen. Ellers returneres katalogkoden 0,1,2 eller 3 i AL.

```
**************************************************
*                                                *
*  Funksjon 20:   Les sekvensielt                *
*                                                *
**************************************************
*                                                *
*          Les neste filsektor                   *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 14H                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL =  0 hvis sektor lest OK      *
*                    <> 0 hvis slutt på filen    *
*                                                *
**************************************************
```

Les-sekvensielt funksjonen leser neste logiske filsektor fra filen til systemets lese/skrive buffer. Spesifisert FKB må tidligere blitt aktivisert ved et åpne-fil kall eller et lag-fil kall (funksjon 15 og 22). Sektoren som leses er gitt ved den løpende sektor telleren, "ls", og filsegment telleren "sm" i FKBen. Ved lesing økes "ls" til neste sektorposisjon i det stående filsegmentet. Funksjonen øker også segment telleren til neste filsegment hvis den siste sektoren i et segment leses. Programmer som skal lese en fil sekvensielt trenger derfor bare sette "ls" feltet til 0 i FKBen etter at filen er åpnet.

Funksjon 20 returnerer verdien 0 i AL-registeret hvis en sektor ble lest korrekt til filsystemets lese/skrive buffer i hukommelsen. Hvis ingen flere sektorer finnes på filen (dvs. slutt på filen), returneres et tall forskjellig fra 0 i AL.

```
**************************************************
*                                                *
*  Funksjon 21:   Skriv sekvensielt              *
*                                                *
**************************************************
*                                                *
*          Skriv neste filsektor                 *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 15H                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL =  0 hvis sektor skrevet OK   *
*                  <> 0 hvis platen er full      *
*                                                *
**************************************************
```

Les-sekvensielt funksjonen skriver neste logiske filsektor til filen fra systemets lese/skrive buffer. Spesifisert FKB må tidligere blitt aktivisert ved et åpne-fil kall eller et lag-fil kall (funksjon 15 og 22). Sektoren som skrives er gitt ved den løpende sektor telleren, "ls", og filsegment telleren "sm" i FKBen. Ved skriving økes "ls" til neste sektorposisjon i det stående filsegmentet. Funksjonen øker også segment telleren til neste filsegment hvis den siste sektoren i et segment skrives. Programmer som skal skrive en fil sekvensielt trenger derfor bare sette "ls" feltet til 0 i FKBen etter at filen er åpnet.

Skriveoperasjoner kan finne sted på eksisterende filer. Gamle filsektorer blir da bare overskrevet.

Funksjon 20 returnerer verdien 0 i AL-registeret hvis sektoren ble korrekt skrevet fra filsystemets lese/skrive buffer i hukommelsen. Hvis platelageret er fullt, returneres et tall forskjellig fra 0 i AL.

```
*************************************************
*                                               *
*  Funksjon 22:   Lag fil                       *
*                                               *
*************************************************
*                                               *
*              Lag en ny fil                    *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 16H                         *
*      Register DX: FKB adresse                 *
*      Register DS: FKB adresse                 *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register AL =  0,1,2,3 hvis filen laget  *
*                     OFFH hvis full katalog    *
*                                               *
*************************************************
```

Lag-fil funksjonen virker på samme måte som åpne-fil funksjonen, bortsett fra at filen som er spesifisert i FKBen ikke må finnes fra før på det aktuelle platelageret under stående brukernummer. Filsystemet lager et nytt element i katalogen for den angitte filen. Brukerprogrammet må selv passe på at filen ikke finnes fra før, ellers blir filen duplisert i katalogen. Dette kan gjøres ved å utføre et fjern-fil kall (funksjon 19) foran lag-fil kallet.

Ved uthopp inneholder AL-registeret verdiene 0,1,2 eller 3 hvis filen ble korrekt laget. Hvis fil-katalogen er full, returneres OFFH (255 desimalt) i AL.

Lag-fil kallet har den sideeffekt at den tilhørende FKBen blir fyllt med nødvendig kataloginformasjon for etterfølgende lese- og skriveoperasjoner. En på-følgende åpne-fil funksjon er derfor ikke nødvendig.

```
***************************************************
*                                                 *
*  Funksjon 23:    Endre filnavn                  *
*                                                 *
***************************************************
*                                                 *
*           Bytt navn på en fil                   *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 17H                           *
*      Register DX: FKB adresse                   *
*      Register DS: FKB adresse                   *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Register AL =  0,1,2,3 hvis navn endret    *
*                     OFFH filen ikke finnes      *
*                                                 *
***************************************************
```

Endre-filnavn funksjonen endrer alle forekomstene i katalogen på spesifisert platelager av filnavnet angitt i de første 12 posisjonene (felt "pl" til og med felt "t3") i FKBen til filnavnet lagret i posisjon 16-27 (felt "d0" til "d11") i samme FKB. Platelager feltet i FKBen ("pl") brukes til å velge ut platelager, mens "pl"-koden for den nye filnavnet, posisjon 16 i FKBen, forutsettes å være 0.

Ved uthopp inneholder AL-registeret verdiene 0,1,2 eller 3 hvis filnavnet ble korrekt endret. Hvis filen ikke finnes, returneres OFFH (255 desimalt) i AL.

```
*************************************************
*                                               *
*  Funksjon 24:    Hent innlogginqstabell       *
*                                               *
*************************************************
*                                               *
*  Hent bit-tabell over plater som er logget inn *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register  CL: 18H                        *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register AX =  innlogginqstabell         *
*                                               *
*************************************************
```

Funksjon 24 returnerer TIKOS 16-bit tabell over hvilke platelagre som er logget inn. Bit-tabellen returneres i AX-registeret, der bit 0 tilsvarer plate A, mens bit 15 tilsvarer plate P. En 1-er i en bit-posisjon angir at platen er logget inn, mens 0 betyr det motsatte.

```
**************************************************
*                                                *
*  Funksjon 25:   Hent utvalgt plate             *
*                                                *
**************************************************
*                                                *
*        Returner stående innlogget plate        *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 19H                         *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register  AL =  stående plate (0-15)      *
*                                                *
**************************************************
```

Funksjon 25 returnerer stående platelager i AL-registeret. Platelagre angis ved et tall fra 0 til 15, der 0 tilsvarer plate A, mens 15 er plate P.

Funksjonen blir brukt av programmer som endrer platelager, og som etter kjøring ønsker å vende tilbake til opprinnelig stående plate.

```
***************************************************
*                                                 *
*  Funksjon 26:  Sett lese/skrive adresse         *
*                                                 *
***************************************************
*                                                 *
*   Sett filsystemets lese/skrive-buffer          *
*   offset adresse.                               *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 1AH                           *
*      Register DX: Lese/skrive offset adresse    *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Ingen                                      *
*                                                 *
***************************************************
```

Funksjon 26 brukes til å sette filsystemets lese/skrive-buffer adresse. Lese/skrive-bufferet er det 128-tegns området i hukommelsen som brukes til mellomlagring av data som skal overføres til og fra filer på platelagrene. Bufferet vil inneholde en filsektor FORAN skriveoperasjoner og ETTER lesekall.

Når TIKOS laste opp et program, vil lese- og skrivebufferet alltid være satt til adresse 80H i programmets systemhukommelse.

```
**************************************************
*                                                *
*  Funksjon 27:  Hent BLOKKTABELL adresse        *
*                                                *
**************************************************
*                                                *
*    Returner adressen til bit-tabell over       *
*    ledige og opptatte blokker for stående      *
*    platelager                                  *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register  CL: 1BH                         *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register BX: Blokktabell offset adresse   *
*                   (AX=BX)                      *
*      Register ES: Blokktabell segment adresse  *
*                                                *
**************************************************
```

TIKOS har en tabell i hukommelsen for hvert platelager som angir hvilke blokker på platelageret som er optatte og ledige. Tabellen er bit-orientert, dvs. hver blokk på platen er representert ved ett bit i tabellen. Funksjonen returnerer tabell-adressen for stående plate i ES:BX registerparet.

Funksjonen kan brukes av spesial-programmer som trenger å vite hvor mye ledig plass som er igjen på platelageret. Vanlige programmer trenger ikke bruke funksjonen.

Blokktabellen er forøvrig beskrevet mer i avsnittet om INN/UT modulen TIUM88.

```
***************************************************
*                                                 *
*  Funksjon 28:  Skrivebeskytt plate              *
*                                                 *
***************************************************
*                                                 *
*    Beskytt stående plate mot skriving.          *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register  CL: 1CH                          *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Ingen                                      *
*                                                 *
***************************************************
```

Skrivebeskytt-plate funksjonen beskytter stående plate mot etterfølgende skriveoperasjoner. Hvis senere skriveoperasjoner forsøkes, vil filsystemet skrive ut feilmeldingen

Feil på <p>: KUN lesbar

hvor <p> er platelager, A-P.

Funksjonen brukes lite i TIKOS.

```
**************************************************
*                                                *
*  Funksjon 29:  Hent skrivebeskyttete plater    *
*                                                *
**************************************************
*                                                *
*    Returner bit-tabell over plater som er      *
*    skrivebeskyttet                             *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*       Register  CL: 1DH                        *
*                                                *
* Utgangsverdier:                                *
*                                                *
*       Register AX: Bit-tabell.                 *
*                                                *
**************************************************
```

Funksjon 29 returnerer en 16-bit tabell over skrivebeskyttete platelagre. Tabellen returneres i AX registeret, der bit 0 tilsvarer plate A, mens bit 15 tilsvaret plate P. En 1-er i en bitposisjon angir at platen er skrivebeskyttet, mens 0 betyr det motsatte.

```
**************************************************
*                                                *
*  Funksjon 30:  Sett filattributt               *
*                                                *
**************************************************
*                                                *
*          Sett attributt(er) på en fil          *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 1EH                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register  AL: 0,1,2,3 hvis attributter    *
*                         korrekt satt           *
*                    0FFH hvis filen ikke finnes *
*                                                *
**************************************************
```

Funksjon 30 merker en angitt fil med visse attributter (egenskaper). Attributtene angis som det øverste bit'et (bit 7) i filnavn-posisjonene ("f1".."f8","t1", "t2","t3") i FKBen. Register AL vil ved uthopp inneholde verdiene 0,1,2 eller 3 hvis attributtene ble korrekt lagret på tilsvarende posisjoner i katalogen. Hvis spesifisert fil ikke finnes, returneres 0FFH (255 desimalt) i AL.

De mest brukte er lese/skrive- og systemattributtet. Lese/skrive attributtet lagres som øverste bit i "t1" posisjonen i filkatalogen og brukes til å skrivebeskytte filer. Filer med systemfil attributtet påsatt (lagret som bit 7 i "t2"), vil ikke bli vist på skjermen av kommandotolkerens katalog kommando.

Brukerprogrammer kan definere andre filattributter i FKB posisjonen "f1" til og med "f4". En viss forsiktighet bør dog utvises, fordi disse attributter kan brukes til spesielle formål i framtidige TIKOS versjoner.

```
***************************************************
*                                                 *
*  Funksjon 31:  Hent platetabell adresse         *
*                                                 *
***************************************************
*                                                 *
*    Returner adressen til platelager tabellen    *
*    for stående plate                            *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*       Register  CL: 1FH                          *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*       Register BX: Tabell offset adresse        *
*                    (AX=BX)                      *
*       Register ES: Tabell offset adresse        *
*                                                 *
***************************************************
```

TIKOS har en tabell i hukommelsen for hvert platelager som beskriver platens størrelse og fysiske utforming. Funksjon 31 returnerer tabell-adressen for stående plate i ES:BX registerparet.

Funksjonen kan brukes av programmer som trenger informasjon om platens fysiske egenskaper. Vanlige programmer trenger ikke bruke denne funksjonen.

Plate-tabellen er lagret i TIKOSs INN/UT modul (TIUM88), og er dokumentert i modulbeskrivelsen (avsnitt 2.11).

```
***************************************************
*                                                 *
*  Funksjon 32:  Sett/hent bruker nr.             *
*                                                 *
***************************************************
*                                                 *
*        Sett nytt bruker nummer eller            *
*        hent stående bruker nummer               *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register  CL: 20H                          *
*      Register  DL: 0FFH hvis hent bruker        *
*      Register  DL: brukernr. hvis sett bruker   *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Hent bruker nummer:                        *
*                                                 *
*        Register  AL: Stående bruker nummer      *
*                                                 *
*      Sett bruker nummer:                        *
*                                                 *
*        Ingen                                    *
*                                                 *
***************************************************
```

Funksjon 32 kan brukes til enten å forandre stående brukernummer eller å få tak i stående brukernummer. Hvis register DL er 0FFH (255 desimalt) ved innhopp, returneres stående brukernummer i AL-registeret. Hvis register DL ikke er 255H, brukes verdien i DL til å sette nytt brukernummer (0-15).

```
**************************************************
*                                                *
*  Funksjon 33:    Les vilkårlig sektor          *
*                                                *
**************************************************
*                                                *
*          Les vilkårlig sektor fra fil          *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 21H                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL =  0 hvis sektor lest OK      *
*                    <> 0 hvis sektor ikke finnes *
*                                                *
**************************************************
```

Les-vilkårlig-sektor funksjonen virker på samme måte som les-sekvensielt funksjonen (funksjon 20) bortsett fra at leseoperasjonen kan finne sted på en spesifisert vilkårlig sektor-posisjon på filen. Posisjonen angis som et 18-bits tall i "r0", "r1" og "r2" feltene i FKBen, der "r0" representerer den laveste delen av sektortallet. "r2" feltet av FKBen må ikke være større enn 3.

Feltene "r0","r1" og "r2" kan inneholde en 18 bits sektorposisjon fra 0 til 262143. Enhver sektor på en maksimalt stor fil på over 33 millioner (33554432) tegn kan derved leses. Funksjonen øker ikke sektor telleren "ls" i FKBen med 1 slik les-sekvensielt funksjonen gjør. Påfølgende kall på funksjon 33 vil derfor lese samme sektoren.

Funksjonen beregner og setter alle FKB tellerne ("sm", "st" og "ls") ved hvert kall. Filen kan derved etterpå leses og skrives sekvensielt fra den aksesserte sektorposisjonen. Imidlertid vil den siste sektoren som ble lest randomisert, bli lest om igjen når programmet går over til å lese sekvensielt. En annen måte å lese sekvensielt på, er å bare øke sektoposisjonen gitt ved "r0","r1" og "r2" med 1 etter hver leseoperasjon.

Funksjon 33 returnerer verdien 0 i AL-registeret hvis en sektor ble lest korrekt til filsystemets lese/skrive buffer i hukommelsen. Følgende verdier forskjellig fra 0 returneres i AL-registeret ved feil:

1,4	:	Sektor ikke tidligere skrevet
6	:	Angitt sektor for stor

Returkode 1 og 4 betyr at programmet har forsøkt å lese en uskrevet sektor. Feilkode 6 angir at logisk sektornummer er større enn maksimalt antall sektorer på platelageret (262144 = 32 Mb), dvs. "r2" i FKBen ved innhopp er større enn 3.

```
*************************************************
*                                               *
*  Funksjon 34:    Skriv vilkårlig sektor       *
*                                               *
*************************************************
*                                               *
*        Skriv vilkårlig sektor til fil         *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 22H                         *
*      Register DX: FKB offset adresse          *
*      Register DS: FKB offset adresse          *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register AL =  0 hvis sektor skrevet OK  *
*                    <> 0 hvis feil             *
*                                               *
*************************************************
```

Skriv-vilkårlig sektor funksjonen virker på samme måte som skriv sekvensielt funksjonen (21) bortsett fra at data skrives til en spesifisert vilkårlig sektor på filen. Posisjonen angis som et 18-bits tall i "r0", "r1" og "r2" feltene i FKBen, der "r0" representerer den laveste delen av sektortallet. "r2" feltet av FKBen må ha verdi mindre enn 4. Funksjonen reserverer også ny plass på platelageret hvis spesifisert sektor ikke før har vært skrevet.

Feltene "r0"-"r1"-"r2" kan inneholde et sektorposisjon tall fra 0 til 262143. Enhver sektor på en maksimalt stor fil på over 33 millioner (33554432 = 32Mb) tegn kan derved skrives. Funksjonen øker ikke sektor telleren "ls" i FKBen med 1 slik skriv-sekvensielt funksjonen gjør. Påfølgende kall på funksjon 34 vil derfor skrive samme sektoren.

Funksjonen beregner og setter alle FKB tellerne ("sm", "st" og "ls") ved hvert kall. Filen kan derved etterpå leses og skrives sekvensielt fra den aksesserte sektorposisjonen. Imidlertid vil den siste sektoren som ble skrevet ved funksjon 34, bli skrevet om igjen når programmet går over til å skrive sekvensielt. En annen måte å skrive sekvensielt på, er å bare øke sektorposisjonen gitt ved feltene "r0"-"r1"-"r2" med 1 etter hvert funksjon 34 kall.

Funksjon 34 returnerer verdien 0 i AL-registeret hvis en sektor ble korrekt skrevet fra filsystemets lese/skrive buffer i hukommelsen. Følgende verdier forskjellig fra 0 returneres i AL-registeret ved feil:

5 : Filkatalogen er full
6 : Angitt sektor for stor

Returkode 5 betyr at katalogen på platen ikke har plass til mer filinformasjon. Feilkode 6 angir at logisk sektornummer er større enn maksimalt antall sektorer på platelageret (262144 = 32Mb), dvs. "r2" i FKBen ved innhopp er større enn 3.

```
*************************************************
*                                               *
*  Funksjon 35:    Beregn fil størrelse         *
*                                               *
*************************************************
*                                               *
*        Beregn størrelsen på en fil i          *
*        antall sektorer                        *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 23H                         *
*      Register DX: FKB offset adresse          *
*      Register DS: FKB segment adresse         *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      "R0"-"R1"-"R2" feltene i FKBen satt      *
*                                               *
*************************************************
```

Beregn-fil-størrelse funksjonen returnerer antallet sektorer filen består av i "R0","R1" og "R2" feltene i FKBen. FKBen må inneholde et entydig filnavn som brukes til å søke i filkatalogen. Returnert fil størrelse er sektor adressen til sektoren som etterfølger den siste sektoren på filen. Hvis "R2" feltet er 04 ved uthopp, er filen på maksimal størrelse, dvs. 262624 filsektorer (= 32 Mb). Hvis "R2" er mindre enn 4, vil "R0"-"R1"-"R2" feltene (18 bits) inneholde filstørrelsen.

Data kan skjøtes på en eksisterende fil ved å kalle funksjon 35 for å sette "R0"-"R1"-"R2" feltene, for deretter å utføre en sekvens av skriv-vilkårlig-sektor kall (funksjon 34) fra den beregnede sektoradressen.

Beregnet filstørrelse er en logisk størrelse som er lik den fysiske størrelsen på en fil som er skrevet sekvensielt. Hvis filen er laget ved hjelp av skriv-vilkårlig-sektor operasjoner med "datahull" i filen, vil beregnet filstørrelse være større en fysisk størrelse. Eksempelvis vil en fil der bare den maksimalt siste sektoren (nr. 262143) er skrevet få beregnet filstørrelse på 262144 sektorer, mens filen fysisk kun opptar en blokk på platelageret.

```
***********************************************
*                                             *
*  Funksjon 36:   Beregn vilkårlig sektor     *
*                                             *
***********************************************
*                                             *
*        Beregn sektor nummer til neste sektor *
*        som skal leses eller skrives         *
*                                             *
***********************************************
*                                             *
* Inngangsparametre:                          *
*                                             *
*      Register CL: 24H                       *
*      Register DX: FKB offset adresse        *
*      Register DS: FKB segment adresse       *
*                                             *
* Utgangsverdier:                             *
*                                             *
*      "R0"-"R1"-"R2" feltene i FKBen satt    *
*                                             *
***********************************************
```

Beregn-vilkårlig-sektor funksjonen beregner det logiske sektornummeret for neste sektor på en fil som har vært aksessert ved hjelp av sekvensielle operasjoner. Funksjonen kan brukes på to måter.

For det første kan det ofte være nyttig å først lese en fil sekvensielt for å undersøke forskjellig "nøkkel"-felt. Etter som hvert nøkkelfelt finnes, kan funksjon 36 kalles for å lagre unna sektorposisjonen for nøkkelfeltet. De forskjellige nøkkelposisjonene kan lagres i en tabell for siden å kunne brukes til å behandle nøkkel-feltene direkte på filen ved hjelp å vilkårlig-sektor filoperasjoner.

En annen bruk av funksjon 36 er ved overgang fra sekvensielle filoperasjoner til vilkårlig-sektor operasjoner. En fil kan således leses og skrives sekvensielt opp til et bestemt punkt, for deretter å behandles randomisert fra det bestemte punktet.

```
*************************************************
*                                               *
*  Funksjon 37:   Logg ut plater                *
*                                               *
*************************************************
*                                               *
*        Logg ut spesifiserte platelagre        *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 25H                         *
*      Register DX: Plate vektor                *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Register  AL: 0                          *
*                                               *
*************************************************
```

Logg-ut-plater funksjonen brukes til å logge ut bestemte platelagre fra filsystemet. Ved innhopp inneholder DX-registeret et 16-bit vektor som spesifiserer hvilke plater som skal logges ut. Bit-0 av DX-registeret satt til 1 betyr at plate A skal logges ut, mens bit-15 av DX tilsvarer plate P.

AL-registeret blir satt til 0 ved uthopp.

```
***************************************************
*                                                 *
*  Funksjon 40:    Skriv vilkårlig sektor med     *
*                  NULL-stilling                  *
*                                                 *
***************************************************
*                                                 *
*     Skriv vilkårlig sektor til fil, fyll        *
*     nye datablokker med 0 før de skrives        *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 28H                           *
*      Register DX: FKB offset adresse            *
*      Register DS: FKB segment adresse           *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Register AL =  0 hvis sektor skrevet OK    *
*                    <> 0 hvis feil               *
*                                                 *
***************************************************
```

Funksjon 40 virker på samme måte som skriv-vilkårlig-sektor funksjonen (nr. 34) bortsett fra at nye datablokker på platelageret som brukes av filen, fylles med nuller før data skrives.

```
*************************************************
*                                               *
*  Funksjon 45:    Sett feil-modus              *
*                                               *
*************************************************
*                                               *
*      Sett filsystem feil modus.               *
*                                               *
*************************************************
*                                               *
* Inngangsparametre:                            *
*                                               *
*      Register CL: 2DH                         *
*      Register DL: 0FFH eller 0FEH             *
*                                               *
* Utgangsverdier:                               *
*                                               *
*      Ingen                                    *
*                                               *
*************************************************
```

Funksjon 45 setter filsystemets feilmodus. Feilmodusen bestemmer hvordan TIKOS oppfører seg ved disk feil. I normal modus, vil TIKOS ved disk feil skrive ut en av følgende fire meldinger på skjermen:

1) Feil på <p>: Les/skriv
2) Feil på <p>: Gal plate
3) Feil på <p>: Fil KUN lesbar
4) Feil på <p>: KUN lesbar

<p> = plate, A,B,..P

Brukeren kvitterer på feilmeldingen ved å trykke en tast, og det kjørende programmet blir stoppet.

Hvis filsystemet er satt i såkalt RETUR modus, vil ingen melding komme opp på skjermen, men en feilstatus vil bli returnert til det kjørende programmet. Feilstatusen består i at AL-registeret ved uthopp blir satt til verdien 0FFH, og AH-registeret vil bli satt til en av følgende feilkoder:

01 - Ved "les/skriv" feil
04 - Ved "gal plate" feil
03 - Ved "fil KUN lesbar" feil
02 - Ved "KUN lesbar" feil

Funksjon 45 setter filsystemets feilmodus fra verdien av DL-registeret ved innhopp. Hvis DL-registeret er satt til 0FFH, blir "vanlig" feilmodus satt. Hvis DL er forskjellig fra 0FFH (bruk 0FEH), vil filsystemet bli satt i RETUR modus.

```
*****************************************************
*                                                   *
*  Funksjon 47:    Start opp et annet program       *
*                                                   *
*****************************************************
*                                                   *
*     Avslutt kjørende program og start opp         *
*     et nytt.                                      *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*      Register CL: 2FH                             *
*      Inn/ut buffer inneholder kommandolinje       *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*      Ingen                                        *
*                                                   *
*****************************************************
```

Funksjon 47 avslutter det kjørende programmet og starter opp et nytt. Filsystemets inn/ut buffer må ved innhopp være fylt med ønsket kommandolinje. Kommandolinjen må være lagret som en sammenhengende tekst streng avsluttet med 0, som vist nedenfor.

```
   Inn/ut buffer:     0  1 ...  n    n+1
                     ------------------------
                     ! <Tekst streng> !  0  !
                     ------------------------
```

```
***************************************************
*                                                 *
*  Funksjon 50:   Utfør TIUM88 funksjon           *
*                                                 *
***************************************************
*                                                 *
*     Utfør funksjon i Tiki Inn/Ut modul TIUM88   *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 32H                           *
*      Register DX: TIUMKB offset adresse         *
*      Register DS: TIUMKB segment adresse        *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      TIUM88 retur                               *
*                                                 *
***************************************************
```

Funksjon 50 gjør det mulig for programmer å utføre basis inn/ut funksjoner i TIUM88 modulen. Ved innhopp må register DX inneholde adressen til en kontrollblokk i hukommelsen, TIUMKB, med følgende innhold:

```
                 0    1    2    3    4
                ---------------------------------
TIUMKB:         ! FUNK !   CX   !   DX    !
                ---------------------------------

                 FUNK = TIUM88 funksjon nr.
                        02 = tastatur status
                        03 = tastatur inn
                        04 = skjerm ut
                        etc.
                 CX   = TIUM88-funksjon CX-register
                 DX   = TIUM88-funksjon DX-register
```

Funksjonen 50 brukes mest for skjerm inn/ut operasjoner.

```
****************************************************
*                                                  *
*  Funksjon 51:  Sett lese/skrive segment adresse  *
*                                                  *
****************************************************
*                                                  *
*   Sett filsystemets lese/skrive-buffer segment   *
*   adresse.                                       *
*                                                  *
****************************************************
*                                                  *
* Inngangsparametre:                               *
*                                                  *
*      Register CL: 33H                            *
*      Register DX: Lese/skrive segment adresse    *
*                                                  *
* Utgangsverdier:                                  *
*                                                  *
*      Ingen                                       *
*                                                  *
****************************************************
```

Funksjon 51 brukes til å sette filsystemets lese/skrive-buffer segment-adresse. Lese/skrive-bufferet er det 128-tegns området i hukommelsen som brukes til mellomlagring av data som skal overføres til og fra filer på platelagrene. Bufferet vil inneholde en filsektor FORAN skriveoperasjoner og ETTER lesekall.

Når TIKOS laste opp et program, vil lese- og skrivebufferet alltid være i adresse 80H i programmets datagruppe. Funksjon 51 trenger således bare benyttes for programmer som skal foreta fil lese- og skrive-operasjoner utenfor programmets egen datagruppe.

```
*****************************************************
*                                                   *
*  Funksjon 52:  Returner lese/skrive adresse       *
*                                                   *
*****************************************************
*                                                   *
*   Returner filsystemets inn/ut buffer             *
*   adresse.                                        *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*      Register CL: 34H                             *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*      Register AX: Inn/ut buffer offset adresse    *
*      Register ES: Inn/ut buffer segment adresse   *
*                                                   *
*****************************************************
```

Funksjon 52 brukes til å lese av TIKOS nåværende fil inn/ut buffer adresse. Funksjonen kan være nyttig for programmoduler som trenger å endre inn/ut buffer adressen, men som senere ønsker å sette buffer adressen tilbake den opprinnelige verdien.

```
****************************************************
*                                                  *
*  Funksjon 53:    Hent maksimal hukommelse        *
*                                                  *
****************************************************
*                                                  *
*     Undersøk hvor mye hukommelse som maksimalt   *
*     er tilgjengelig, ikke tildel ("alloker")     *
*                                                  *
****************************************************
*                                                  *
* Inngangsparametre:                               *
*                                                  *
*      Register CL: 35H                            *
*      Register DX: HKB offset adresse             *
*      Register DS: HKB segment adresse            *
*                                                  *
* Utgangsverdier:                                  *
*                                                  *
*      Register AL =  0    hvis hukommelse finnes  *
*                  =  0FFH hvis hukommelse mangler *
*                                                  *
****************************************************
```

Funksjon 53 undersøker hvor mye hukommelse som er ledig for programmet. DX-registeret inneholder ved innhopp adressen til en hukommelses kontroll blokk, HKB, med følgende format:

```
              0    1    2    3    4
             -----------------------------
HKB:         !  BASE   !  LENGDE  ! EXT !
             -----------------------------

             BASE   = hukommelses segment adresse
             LENGDE = størrelse i antall paragrafer
                      1 paragraf = 16 hukommelses celler
             EXT    = ikke brukt, må være tilstede
```

Funksjon 53 returnerer den største sammenhengende hukommelses-blokken som er mindre enn angitt LENGDE i HKB'en. Funksjonen fyller inn BASE- og LENGDE feltet i HKB'en for funnet hukommelse. Funksjonen returnerer 0 i AL-registeret hvis hukommelse finnes. Ellers returneres 0FFH i AL.

Funksjon 53 tildeler <u>ikke</u> hukommelse. Funksjonen bare returnerer hvor mye som er ledig.

```
*****************************************************
*                                                   *
*  Funksjon 54:  Hent absolutt maksimal hukommelse *
*                                                   *
*****************************************************
*                                                   *
*     Undersøk hvor mye hukommelse som maksimalt    *
*     er tilgjengelig fra en angitt base, ikke      *
*     tildel ("alloker")                            *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*      Register CL: 36H                             *
*      Register DX: HKB offset adresse              *
*      Register DS: HKB segment adresse             *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*      Register AL =  0    hvis hukommelse finnes   *
*                  =  OFFH hvis hukommelse mangler *
*                                                   *
*****************************************************
```

Funksjon 54 undersøker hvor mye hukommelse fra en fast base som er ledig for programmet. DX-registeret inneholder ved innhopp adressen til en hukommelses kontroll blokk, HKB, med følgende format:

```
           0    1     2    3     4
          ------------------------------
HKB:      !   BASE   !  LENGDE  ! EXT !
          ------------------------------

          BASE   = hukommelses segment adresse
          LENGDE = størrelse i antall paragrafer
                   1 paragraf = 16 hukommelses celler
          EXT    = ikke brukt, må være tilstede
```

Funksjon 54 returnerer den største sammenhengende hukommelses-blokken fra spesifisert BASE som er mindre enn LENGDE feltet i HKB'en. Funksjonen fyller inn LENGDE feltet i HKB'en for funnet hukommelse. Funksjonen returnerer 0 i AL-registeret hvis hukommelse finnes. Ellers returneres OFFH i AL.

Funksjon 54 tildeler ikke hukommelse. Funksjonen bare returnerer hvor mye som er ledig.

```
*****************************************************
*                                                   *
*  Funksjon 55:   Tildel hukommelse                 *
*                                                   *
*****************************************************
*                                                   *
*     Tildel ("alloker") hukommelse til programmer *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*      Register CL: 37H                             *
*      Register DX: HKB offset adresse              *
*      Register DS: HKB segment adresse             *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*      Register AL =  0    hvis hukommelse finnes   *
*                  =  0FFH hvis hukommelse mangler *
*                                                   *
*****************************************************
```

DX-registeret inneholder ved innhopp til funksjon 55 adressen til en hukommelses kontroll blokk, HKB, med følgende format:

```
              0    1    2    3    4
             ------------------------------
  HKB:       !   BASE    !  LENGDE  ! EXT !
             ------------------------------

             BASE   = hukommelses segment adresse
             LENGDE = størrelse i antall paragrafer
                      1 paragraf = 16 hukommelses celler
             EXT    = ikke brukt, må være tilstede
```

Funksjon 55 tildeler ("allokerer") hukommelse med lengde lik LENGDE feltet i HKB'en. Funksjonen fyller inn BASE-feltet i HKB'en for funnet hukommelse. Funksjonen returnerer 0 i AL-registeret hvis hukommelse finnes. Ellers returneres 0FFH i AL.

```
**************************************************
*                                                *
*  Funksjon 56:  Tildel absolutt hukommelse      *
*                                                *
**************************************************
*                                                *
*     Tildel ("alloker") hukommelse til programmet *
*     på fast adresse                            *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 38H                          *
*      Register DX: HKB offset adresse           *
*      Register DS: HKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AL =  0    hvis hukommelse finnes *
*                  =  0FFH hvis hukommelse mangler *
*                                                *
**************************************************
```

DX-registeret inneholder ved innhopp til funksjon 55 adressen til en hukommelses kontroll blokk, HKB, med følgende format:

```
              0    1     2    3    4
             ------------------------------
HKB:         !   BASE   !  LENGDE  ! EXT !
             ------------------------------

             BASE   = hukommelses segment adresse
             LENGDE = størrelse i antall paragrafer
                      1 paragraf = 16 hukommelses celler
             EXT    = ikke brukt, må være tilstede
```

Funksjon 56 tildeler ("allokerer") hukommelse på en fast adresse spesifisert i BASE feltet i HKB'en og med lengde lik LENGDE feltet. Funksjonen returnerer 0 i AL-registeret hvis hukommelse finnes. Ellers returneres 0FFH i AL.

```
*****************************************************
*                                                   *
*  Funksjon 57:   Frigi hukommelse                  *
*                                                   *
*****************************************************
*                                                   *
*     Frigi tildelt hukommelse                      *
*                                                   *
*****************************************************
*                                                   *
* Inngangsparametre:                                *
*                                                   *
*      Register CL: 39H                             *
*      Register DX: HKB offset adresse              *
*      Register DS: HKB segment adresse             *
*                                                   *
* Utgangsverdier:                                   *
*                                                   *
*      Register AL =  0    hvis tildelt fra før     *
*                   =  0FFH ikke tidligere tildelt  *
*                                                   *
*****************************************************
```

DX-registeret inneholder ved innhopp til funksjon 57 adressen til en hukommelses kontroll blokk, HKB, med følgende format:

```
              0     1     2     3     4
              -----------------------------
   HKB:       !   BASE    !  LENGDE  ! EXT !
              -----------------------------
```

```
BASE   = hukommelses segment adresse
LENGDE = størrelse i antall paragrafer
         1 paragraf = 16 hukommelses celler
EXT    = kontroll ord
```

Funksjon 57 frigir tildelt hukommelse. Hvis EXT-feltet i HKB'en er lik 0, frigis hukommelse med angitt BASE og LENGDE i HKB'en. Hvis EXT er lik 0FFH, frigis all hukommelse som er tildelt programmet, untatt den hukommelsen som opptas av programmet selv.

```
***************************************************
*                                                 *
*  Funksjon 58:   Frigi all hukommelse            *
*                                                 *
***************************************************
*                                                 *
*     Frigi all tildelt hukommelse, intitier      *
*     hukommelses-tabeller                        *
*                                                 *
***************************************************
*                                                 *
* Inngangsparametre:                              *
*                                                 *
*      Register CL: 3AH                           *
*                                                 *
* Utgangsverdier:                                 *
*                                                 *
*      Ingen                                      *
*                                                 *
***************************************************
```

Funksjon 58 frigir all tildelt hukommelse, inklusive hukommelsen hvor det kallende programmet ligger. Funksjonen initierer samtidig alle interne hukommelses-tabeller i TIKOS.

```
**************************************************
*                                                *
*  Funksjon 59:    Program last                  *
*                                                *
**************************************************
*                                                *
* Last et nytt program inn i hukommelsen         *
*                                                *
**************************************************
*                                                *
* Inngangsparametre:                             *
*                                                *
*      Register CL: 3BH                          *
*      Register DX: FKB offset adresse           *
*      Register DS: FKB segment adresse          *
*                                                *
* Utgangsverdier:                                *
*                                                *
*      Register AX: 0FFFFH hvis feil             *
*                   Program systemområde segment *
*                   adresse hvis OK lastet       *
*                                                *
**************************************************
```

Funksjon 59 laster en CMD-type fil inn i hukommelsen og forbereder programmet for kjøring. Hukommelse blir tildelt det nye programmet, og program systemområdet blir satt opp. Programfilen gitt i FKB'en må være åpnet før funksjon 59 kalles.

Ved uthopp blir register AX satt til 0FFFFH ved feil. Ellers vil AX inneholde segmentadressen til det nye programmets systemområde. Adressene til de andre programgruppene fra CMD filen finnes i systemområdet.

2.10 TIKOS spesial funksjoner.

TIKOS inneholder seks spesialfunksjoner for direkte kommunikasjon med hovedprosessoren Z80. Funksjonene gjør det mulig å utføre inn/ut operasjoner direkte mot Z80 porter, samt å lese/skrive direkte inn i Z80's hukommelse. Funksjonene utføres ved å sette opp passende verdier i 8088's registre, og dernest utføre en programstyrt "interrupt" (INT) maskinkode-instruksjon.

2.10.1 Les Z80 dataport.

8088 instruksjon:	INT	0A0H	(160)
Inngangsparameter:	DX	=	8 bit Z80 portadresse
Utgangsparameter:	AL	=	port verdi

Denne funksjonen gjør det mulig å lese en Z80 dataport direkte. Z80's portnummer spesifiseres i DX-registeret, og lest portverdi blir returnert i AL.

2.10.2 Skriv Z80 dataport.

8088 instruksjon:	INT	0A1H	(161)
Inngangsparametre:	DX	=	8 bit Z80 portadresse
	AL	=	port verdi
Utgangsparametre:	Ingen		

Denne funksjonen gjør det mulig å skrive til en Z80 dataport direkte. Z80's portnummer spesifiseres i DX-registeret, og verdien som skal ut på porten settes i AL.

2.10.3 Utfør Z80 subrutine.

8088 instruksjon:	INT	0A2H (162)
Inngangsparametre:	AL	= Z80 A-register
	CX	= Z80 BC-register
	DX	= Z80 DE-register
	BX	= Z80 HL-register
	SI	= Z80 subrutine adresse
Utgangsparametre:	AL	= Z80 A-register
	CX	= Z80 BC-register
	DX	= Z80 DE-register
	BX	= Z80 HL-register

Denne funksjonen gjør det mulig å utføre en Z80 subrutine på fast adresse. SI-registeret inneholder Z80 adressen ved innhopp, og subrutinens registerparametre overføres i tilsvarende 8088 registre som vist ovenfor. Z80-rutinens returparametre blir ved uthopp kopiert inn i de tilsvarende 8088 registre.

2.10.4 Hent Z80 segment.

8088 instruksjon:	INT	0A3H (163)
Inngangsparametre:	Ingen	
Utgangsparameter:	AX	= Z80 segment base (0C000H)

Z80's 64K lokale hukommelse kan leses/skrives fra 8088 prosessoren. "Interrupt"-funksjon 163 returnerer segment basen sett fra 8088 til Z80 hukommelsen. Når dette skrives er Z80 segmentbasen lik 0C000H, men dette kan endres i senere versjoner.

2.10.5 Flytt 8 bit hukommelse.

8088 instruksjon:	INT	0A4H (164)
Inngangsparametre:	DS	= FRA hukommelse segment
	SI	= FRA hukommelse offset
	ES	= TIL hukommelse segment
	DI	= TIL hukommelse offset
	CX	= Antall 8 bit celler
Utgangsparametre:	Ingen	

Denne funksjonen gjør det mulig å flytte en blokk 8 bit hukommelse til/fra Z80's lokale hukommelse fra/til 8088 hukommelsen. Flyttingen kan også kun foregå internt på Z80's lokale hukommelse.

2.10.6 Flytt 16 bit hukommelse.

8088 instruksjon:	INT	0A5H (165)
Inngangsparametre:	DS	= FRA hukommelse segment
	SI	= FRA hukommelse offset
	ES	= TIL hukommelse segment
	DI	= TIL hukommelse offset
	CX	= Antall 16 bit celler
Utgangsparametre:	Ingen	

Denne funksjonen gjør det mulig å flytte en blokk 16 bit hukommelse til/fra Z80's lokale hukommelse fra/til 8088 hukommelsen. Flyttingen kan også kun foregå internt på Z80's lokale hukommelse.

2.11 Inn/Ut modul (TIUM88)

Tiki Inn/Ut Modulen, TIUM88, inneholder 21 grunnleggende funksjoner som brukes av det øvrige operativsystem (TIFS88). Innholdet i dette avsnittet er beregnet for spesielt interesserte, og behøver ikke leses for å kunne lage "vanlige" TIKOS programmer.

TIUM88 funksjonene faller i 3 klasser, filsystem funksjonene, tegn inn/ut funksjoner, og en funksjon for spesifisering av tilgjengelig hukommelse. TIUM88 modulen kopler TIKOS til Z80's elektronikknære programvare. TIUM88 modulen omformer de fleste innkommende funksjoner til kall på tilsvarende funksjoner i Z80's TIUM modul.

TIUM88 modulen inneholder en hopptabell for de enkelte funksjonene i starten på modulen. Hopptabellen starter på fast adresse 2500 Hex fra starten av selve TIKOS operativsystemet. Hver funksjon bruker 3 hukommelsesceller i tabellen, 1 celle for hopp-instruksjonskoden ("JMP") og 2 celler til den relative hoppadressen i TIUM88 modulen. Hopptabellens format er vist i tabell 2.5.

Brukerprogrammer kan i sjeldnere tilfeller kalle TIUM88 funksjoner direkte. Dette gjøres ved hjelp av TIKOS funksjon 50.

Adresse	Nr.	Innhold	Beskrivelse
TIUM88+0	0	JMP START	;Kommer her etter at ;strømmen er slått på
TIUM88+3	1	JMP EXIT	;Kommer her når et program ;skal avslutte
TIUM88+6	2	JMP TASTST	;Tastatur status, sjekker om ;tast er trykket
TIUM88+9	3	JMP TASTINN	;Leser neste tegn fra tasta- ;turet, venter til tast er ;trykket.
TIUM88+12	4	JMP SKJERMUT	;Skriver 1 tegn på skjermen
TIUM88+15	5	JMP SKRIVUT	;Sender 1 tegn til skriver
TIUM88+18	6	JMP PAPBÅNDI	;Skriv tegn på papirbånd, ;leser fra serieport P2
TIUM88+21	7	JMP PAPBÅNDU	;Les tegn fra papirbånd, ;skriver til serieport P2
TIUM88+24	8	JMP HJEM	;Gå til spor 0 på platen
TIUM88+27	9	JMP VELGPL	;Velg ut en bestemt plate
TIUM88+30	10	JMP SETTSP	;Velg et bestemt platespor
TIUM88+33	11	JMP SETTSEK	;Velg en bestemt fysisk ;sektor på platen
TIUM88+36	12	JMP SETTOADR	;Sett lese/skrive offset ;adresse
TIUM88+39	13	JMP LES	;Les fysisk sektor fra plate
TIUM88+42	14	JMP SKRIV	;Skriv fysisk sektor
TIUM88+45	15	JMP SKRSTAT	;Se om skriver er klar til å ;motta et nytt tegn
TIUM88+48	16	JMP SEKTRANS	;Transformer logisk sektor ;til fysisk sektor nummer
TIUM88+51	17	JMP SETTSADR	;Sett lese/skrive segment ;adresse
TIUM88+54	18	JMP SEGTAB	;Returner adresse til ;hukommelses-tabell
TIUM88+57	19	JMP HIUC	;Hent inn/ut celle
TIUM88+60	20	JMP SIUC	;Sett inn/ut celle

Tabell 2-5. TIUM88 hopptabell.

2.9.1 TIUM88 funksjonene

Nedenfor følger en kort beskrivelse av de enkelte TIUM88 funksjonene. Funksjonene er identiske med standard CP/M-86 BIOS funksjoner.

Funksjon 0: START - adresse TIUM88+0

Funksjonen setter opp TIKOS systemet etter at strømmen er slått på. Funksjonen utfører en del grunnleggende oppsetting av elektronikken og eksekveres kun ved første gangs oppstart av TIKOS.

Funksjon 1: EXIT - adresse TIUM88+3

EXIT funksjonen utføres når et program har avsluttet ved hjelp av TIKOS funksjon 0. Funksjonen sørger for å reaktivere Z80 prosessoren.

Funksjon 2: TASTS - adresse TIUM88+6

Utparameter: AL-reg = 0FFH hvis tegn klart
AL-reg = 0 hvis tegn ikke er klart

TASTS sjekker om en tast er trykket på tastaturet. Funksjonen returnerer 0 i AL-registeret hvis ingen tast er trykket. Hvis nytt tegn er klart fra tastaturet returneres 0FFH i AL.

Funksjon 3: TASTINN - adresse TIUM88+9

Utparameter: AL-reg = Neste tegn

TASTINN returnerer neste tegn fra tastaturet i AL-registeret. Funkjonen venter til en tast er trykket.

Funksjon 4: SKJERMUT - adresse TIUM88+12

Innparameter: CL-reg = Skjerm tegn

SKJERMUT funksjonen skriver tegnet i CL-registeret ut på videoskjermen.

Funksjon 5: SKRIVUT - adresse TIUM88+15

Innparameter: CL-reg = Skriver tegn

SKRIVUT funksjonen sender tegnet i CL-registeret til tilkoplet skriver. Skriveren kan være tilkoplet TIKI datamaskinen enten via serie- eller parallell port.

Funksjon 6: PAPBÅNDU - adresse TIUM88+18

Innparameter: CL-reg = Tegn

Reservert for å skrive et tegn til papirbånd. Brukes til å skrive til serieport P2.

Funksjon 7: PAPBÅNDI - adresse TIUM88+21

Utparameter: AL-reg = Lest tegn

Reservert for å lese et tegn fra papirbånd. Brukes til å lese fra serieport P2.

Funksjon 8: HJEM - adresse TIUM88+24

HJEM funksjonen angir at lese/skrive hodet for disketter (eller andre platelagre) skal flyttes til spor 0. Den fysiske operasjonen vil som oftest ikke utføres før lesing eller skriving foretas.

Funksjon 9: VELGPL - adresse TIUM88+27

Innparametre: CL-reg = Plate nummer
DL-reg = 0 hvis første aksess på platen etter systemopplast

Utparameter: BX-reg = adresse til platelager tabell for angitt plate
BX-reg = 0 hvis platen ikke finnes

VELGPL velger ut et bestemt TIKOS platelager, 0-15, for etterfølgende lese/skrive operasjoner. CL-registeret inneholder platelager. Hvis DL-registeret er 0 ved innhopp, er dette første aksessen på platen etter at TIKOS er lastet opp eller etter en tidligere "frigivelse" av platen (funksjon 13 og 37).

Funksjon 10: SETTSP - adresse TIUM88+30

Innparameter: CX-reg = Spor nummer

SETTSP funksjonen velger ut et bestemt spor, gitt i CX-registeret, på utvalgt platelager.

Funksjon 11: SETTSEK - adresse TIUM88+33

Innparameter: CX-reg = Sektor nummer

SETTSEK funksjonen velger ut en bestemt fysisk sektor, gitt i CX-registeret, på utvalgt plate og spor.

Funksjon 12: SETTOADR - adresse TIUM88+36

Innparameter: CX-reg = Lese/skrive offset adresse

SETTOADR funksjonen setter systemets lese- og skrive offset adresse for filfunksjoner. All data som leses fra platelageret vil bli lagt ut i hukommelsen gitt ved denne adressen. Tilsvarende vil data som skrives til platelagrene tas fra denne hukommelsesadressen. TIUM88 overfører alltid blokker på 128 tegn til og fra platelagrene.

Funksjon 13: LES - adresse TIUM88+39

Utparameter: AL-reg = 0 hvis sektor lest OK
AL-reg = -1 hvis lesefeil

LES funksjonen leser en TIKOS sektor (=128 tegn) fra angitt plate/spor/sektor til hukommelsen gitt av spesifisert lese/skrive adresse. Funksjonen returnerer 0 i AL-registeret hvis lesingen gikk bra. Hvis lesefeil returneres 0FFH.

Funksjon 14: SKRIV - adresse TIUM88+42

Innparameter: CL-reg = 0 hvis vanlig skriv
CL-reg = 1 hvis skriving til fil-katalogen
CL-reg = 2 hvis skriving til første sektor på en ny datablokk som ikke inneholder gyldig data

Utparameter: AL-reg = 0 hvis sektor skrevet OK
AL-reg = -1 hvis skrivefeil

SKRIV funksjonen skriver en TIKOS sektor (=128 tegn) fra hukommelsen gitt ved spesifisert lese/skrive adresse ut på angitt adresse ut platelager/spor/sektor. Funksjonen returnerer 0 i AL-registeret hvis skrivingen gikk bra. Ved skrivefeil returneres 0FFH i AL. Ved innhopp inneholder CL-registeret verdiene 0,1 eller 2. 0 betyr vanlig skriv, 1 betyr skriving til filkatalogen, mens 2 betyr skriv til første sektor på en datablokk som ikke er i bruk. Denne informasjonen brukes i TIUM88 til å effektivisere skriving til platelagre der fysisk sektor størrelse er større enn 128 (for dobbel tetthet disketter er fysisk sektor 512 tegn). En effektiv metode for å minimalisere fysiske skrive-operasjoner ved store fysiske sektorstørrelser (såkalt "deblocking"), er av stor betydning for filsystemets hurtighet.

Funksjon 15: SKRSTAT - adresse TIUM88+45

Utparameter: AL-reg = -1 hvis skriver klar til å motta et nytt tegn
AL-reg = 0 hvis skriver IKKE er klar til å motta tegn

SKRSTAT funksjonen tester om tilkoplet skriver er klar til å motta et nytt tegn. Hvis skriveren er klar, returneres 0FFH i AL-registeret. Ellers returneres 0 i AL.

Funksjon 16: SEKTRANS - adresse TIUM88+48

Innparametre: CX-reg = Logisk sektor
DX-reg = Adresse til transformasjons-tabell
Utparameter: BX-reg = Fysisk sektor

SEKTRANS funksjonen omformer fra logisk sektor nummer til fysisk sektor ved hjelp av en konverteringstabell. CX inneholder logisk sektor nummer ved innhopp. Dette tallet brukes som en indeks i en konverteringstabell gitt ved DX-registeret. Indeksert tabellelement returneres i BX-registeret, som da er korrekt fysisk sektor.

Funksjon 17: SETTSADR - adresse TIUM88+36

Innparameter: CX-reg = Lese/skrive segment adresse

SETTSADR funksjonen setter systemets lese- og skrive segment adresse for filfunksjoner. All data som leses fra platelageret vil bli lagt ut i hukommelsen gitt ved denne adressen. Tilsvarende vil data som skrives til platelagrene tas fra denne hukommelsesadressen. TIUM88 overfører alltid blokker på 128 tegn til og fra platelagrene.

Funksjon 18: SEGTAB - adresse TIUM88+45

Utparameter: BX-reg = adresse til segment tabell over hukommelse

SEGTAB returnerer adressen til segment-tabellen som beskriver tilgjengelig hukommelse. Segment-tabellen er lagret i TIUM88 modulet, og har følgende format:

```
    !-----------------!
 0  !  Antall segm.   !
    !-----------------!
 1  !  Segment 1      !
 2  !    base         !
    !-----------------!
 3  !  Segment 1      !
 4  !   lengde        !
    -------------------
 5  !  Segment 2      !
 6  !    base         !
    !-----------------!
 7  !  Segment 2      !
 8  !   lengde        !
    -------------------
    !      etc.       !
```

Funksjon 19: HIUC - adresse TIUM88+45

Utparameter: AL-reg = inn/ut celle

HIUC funksjonen henter fram aktuell verdi av systemets INN/UT celle som er lagret i Z80's hukommelse. Cellen blir brukt til å kunne dirigere terminal INN/UT enten til innebygget tastatur og tilkoplet videoskjerm eller til spesial-tilkoplet terminal over serielinje. INN/UT cellen er beskrevet under forklaringen til TIKOS funksjon 7 i avsnitt 2.9.

Funksjon 19: SIUC - adresse TIUM88+45

Innparameter: AL-reg = inn/ut celle

SIUC funksjonen setter ny verdi inn i systemets INN/UT celle som er lagret i Z80's hukommelse. Cellen blir brukt til å kunne dirigere terminal INN/UT enten til innebygget tastatur og tilkoplet videoskjerm eller til spesial-tilkoplet terminal over serielinje. INN/UT cellen er beskrevet under forklaringen til TIKOS funksjon 7 i avsnitt 2.9.

2.9.2 Filsystem tabellene.

Hvert platelager i TIKOS er beskrevet i en platetabell for vedkommende plate. Platetabellene (PT) finnes i TIUM88 modulen som returnerer adressen til tabellen ved VELGPL funksjonen (funksjon nr. 9). Platetabellene opptar hver 16 hukommelses-celler og har følgende format:

STR	0	0	0	KBUF	PPAR	SV	BLT
0	2	4	6	8	10	12	14

(Lokasjon nr.)

Platetabell inneholder følgende elementer:

STR — Adressen til sektor-transformasjons-tabellen for denne platen. Tabellen omformer fra logisk til fysisk sektor nummer og brukes av SEKTRANS funksjonen (nr. 17) i TIUM88-modulen. Hvis STR har verdien 0, vil ingen omforming fra logisk til fysisk sektor finne sted. Fysisk sektor vil da være lik logisk sektor. Platelagre som har samme sektoromforming kan dele de samme transformasjons-tabellene.

0 — Brukes til kladdeområder av filsystemet.

KBUF — Adresse til et 128-tegns kladdeområde som brukes ved lesing og skriving av fil-katalogen. Alle platetabellene kan adressere det samme KBUF området.

PPAR — Adresse til plateparameter-tabell, beskrevet nedenfor. Plater med samme plate-parametre kan dele parameter tabeller.

SV — Adresse til en sjekk vektor som brukes av filsystemet til å finne ut om disketter har vært byttet. Det må finnes en sjekk vektor pr. plate. Lengden av SV er gitt i plate-parameter-tabellen (PPAR).

BLT — Adresse til en blokk-allokerings tabell brukt til å holde oversikt over hvilke blokker på platene som er opptatt og ledige. Hvert platelager må ha sin egen allokeringstabell. Lengden av BLT er gitt

i plate-parameter-tabellen (PPAR). Adressen av BLT returneres av TIKOS programfunksjon 27.

Plate-parameter-tabellene (PPAR) opptar 15 hukommelses-celler og har følgende format:

SPS	BSF	BM	SM	MD	MF	A0	A1	SVS	KS
(2)	(1)	(1)	(1)	(2)	(2)	(1)	(1)	(2)	(2)

Hvert tabell-element er på enten 1 eller 2 hukommelses-celler som angitt ovenfor. Parameter-tabellen inneholder følgende elementer:

SPS Antall Sektorer Pr. Spor på platen.

BSF Blokk Skift Faktor. 2 opphøyd i BSF angir hvor mange TIKOS sektorer (128 tegn) som går i hver blokk på platelageret. BSF bestemmer derved blokkstørrelsen på platen. En blokk er den minste plassenheten som kan tildeles filer på platelageret.

BM Blokk Maske. BM+1 angir hvor mange TIKOS sektorer som går i hver blokk.

SM Segment Maske. Bestemt av antall blokker på platen (MD) og blokkstørrelsen (BLS) på følgende måte:

1. Hvis maksimalt antall blokker på platen (MD) er mindre enn 256, er SM bestemt av følgende tabell:

BLS	SM
1024	0
2048	1
4096	3
8192	7
16384	15

2. Hvis MD er større enn 255, bestemmes SM av følgende tabell:

BLS	SM
1024	Ikke lovlig
2048	0
4096	1
8192	3
16384	7

MD — Maksimal antall Datablokker på platen - 1. MD bestemmer også lengden av allokeringsvektoren BLT i platetabellen. Lengden er gitt ved (MD/8)+1.

MF — Maksimalt antall filelementer i filkatalogen på platen - 1. Bestemmer også verdien av A0 og A1 feltene (se nedenfor).

A0 og A1 — Bit tabell over hvor mange blokker som er opptatt av filkatalogen. Bit 7 av A0 tilsvarer blokk 0 på platen, bit 0 av A0 tilsvarer blokk 7. På samme måten representerer A1 blokk 8-15 på platen. En 1-er i en bit-posisjon angir at blokken er opptatt av filkatalogen. Hvert filelement i katalogen opptar 32 tegn. Hvis f.eks. platens blokkstørrelse er på 1024 tegn, vil således en blokk kunne inneholde 32 filelementer. Hvis platen skal ha plass til 64 filer (MF=63) og blokkstørrelsen er 1024, vil 2 blokker måtte reserveres for filkatalogen. Verdien av A0 vil derved være 0C0 Hex, mens A1 vil ha verdi 0. Dette er TIKOS's standardverdier for enkel og dobbel tetthet disketter.

SVS — Sjekk Vektor Størrelse. SVS bestemmes på følgende måte:

1. Hvis platelageret er utskiftbart (f.eks. disketter), settes SVS til (MF+1)/4.
2. Hvis platelageret er fast, settes SVS til 0.

KS — Katalog Start. Angir hvilket spor på platen filkatalogen starter. Normalt vil de ytterste sporene (fra spor 0) av platelagrene brukes til å lagre TIKOS systemet selv. For disketter vil TIKOS normalt oppta 2 eller 3 spor. KS vil da være 2 eller 3.

VEDLEGG A

Feilmeldinger.

Feilmeldinger under program-opplast:

8088?
Finner ikke TIKOS.SYS filen
8088 prosessor mangler
Hukommelse mangler

Feilmeldinger under kjøring:

Feil på <p>: Les/skriv
Feil på <p>: Gal plate
Feil på <p>: Fil KUN lesbar
Feil på <p>: KUN lesbar

<p> = A,B,..P

** <feilmelding> i: ssss:oooo - Programmet stoppet **

<feilmelding> = Divisjon med 0
<feilmelding> = Register overflyt
<feilmelding> = IBM-PC interrupt 10 ulovlig
<feilmelding> = Ulovlig interrupt <nn>

<nn>=hex

VEDLEGG B

Programfunksjons-sammendrag.

Nr.	Funksjonsnavn	Inn parametre	Ut parametre
0	Last opp TIKO	Ingen	Ingen
1	Tastatur inn	Ingen	A=tegn
2	Skjerm ut	E=tegn	Ingen
3	Port inn	Ingen	A=tegn
4	Port ut	E=tegn	Ingen
5	Skriver ut	E=tegn	Ingen
6	Direktr terminal inn/ut	Se definisjon	Se definisjon
7	Hent INN/UT celle	Ingen	A=INN/UT celle
8	Sett INN/UT celle	E=INN/UT celle	Ingeen
9	Skriv tegnsekvens	DX=tegnsekvens-adresse	Ingen
10	Les tastatur linje	DX=linjebuffer	Se definisjon
11	Tastatur status	Ingen	AL=0/0FFH
12	Versjonsnummer	Ingen	AX=0022H
13	Initier filsystemet	Ingen	AL=0/0FFH
14	Velgut plate	DL=plate nr.	Ingen
15	Åpne fil	DX=FKB adresse	AL=Katalog kode
16	Lukk fil	DX=FKB adresse	AL=Katalog kode
17	Søk etter første	DX=FKB adresse	AL=Katalog kode
18	Søk etter neste	Ingen	AL=Katalog kode
19	Fjern fil	DX=FKB adresse	AL=Katalog kode
20	Les sekvensielt	DX=FKB adresse	AL=Feilkode
21	Skriv sekvensielt	DX=FKB adresse	AL=Feilkode
22	Lag fil	DX=FKB adresse	AL=Katalog kode
23	Endre filnavn	DX=FKB adresse	AL=Katalog kode
24	Hent innloggingstabell	Ingen	AX=Tabell
25	Hent utvalgt plate	Ingen	AL=Plate nr.
26	Sett lese/skrive adresse	DX=adresse	Ingen
27	Hent blokktabell adresse	Ingen	BX=adresse
28	Skrivebeskytt plate	Ingen	Ingen
29	Hent skrivebeskyttete plater	Ingen	AX=vektor
30	Sett filattributt	DX=FKB adresse	AL=Katalog kode
31	Hent platetabell adresse	Ingen	BX=Adresse
32	Sett/hent bruker nr.	Se definisjon	Se definisjon
33	Les vilkårlig sektor	DX=FKB adresse	AL=Feilkode
34	Skriv vilkårlig sektor	DX=FKB adresse	AL=Feilkode
35	Beregn filstørrelse	DX=FKB adresse	R0,R1,R2
36	Beregn vi DE=FKB adresse	R0,R1,R2	
37	Logg ut plater	DE=Plate vektor	Ingen
40	Skriv vilkårlig sektor med NULL-stilling	DE=FKB adresse	A=feilkode

VEDLEGG B

Programfunksjons-sammendrag, forts.

Nr.	Funksjonsnavn	Inn parametre	Ut parametre
45	Sett filsystem feilmodus	DL=feilmodus	Ingen
47	Start opp annet program	Se definisjon	Ingen
50	Utfør TIUM88 funksjon	DX=TIUMKB	TIUM retur
51	Set inn/ut buffer segment	DX=segment	Ingen
52	Returner inn/ut adresse	Ingen	ES:BX=adresse
53	Hent maks. hukommelse tilgjengelig	DX=HKB	AL=0/0FFH
54	Hent maks. hukommelse tilgjengelig fra base	DX=HKB	AL=0/0FFH
55	Tildel hukommelse	DX=HKB	AL=0/0FFH
56	Tildel hukommelse fra fast base	DX=HKB	AL=0/0FFH
57	Frigi hukommelse	DX=HKB	AL=0/0FFH
58	Frigi all hukommelse	Ingen	Ingen
59	Programlast	DX=FKB	AL=0FFFFH/ systemområde til program

VEDLEGG C

Eksempel på TIKOS program.

```
CP/M ASM86 1.1  SOURCE: INTV.A86                                    PAGE  1

                    ; VER 1.25 TAB+S9,8 SUB A86 INTV
                    ;
                    ;*** Display all interrupt vectors on TIKI-100 8088 card ***
                    ;
                    ;       ********************************************
                    ;       * Programmer:   Ingar Rune Steinsland      *
                    ;       *               Orkim Data                 *
                    ;       *               Kordahlveien 13            *
                    ;       *               1521 Sperrebotn            *
                    ;       *               Norway                     *
                    ;       *                                          *
                    ;       * Date:         January 23, 1985           *
                    ;       *                                          *
                    ;       * Last update:  January 23, 1985           *
                    ;       *                                          *
                    ;       * (The program is NOT too well documented) *
                    ;       ********************************************
                    ;
                    pagesize 50
                    pagewidth 90
                    ;
                    ;       ******************
                    ;       * ASCII symbols: *
                    ;       ******************
                    ;
   000D             cr              equ     0dh
   000A             lf              equ     0ah
   0020             space           equ     20h
                    ;
                    ;       ****************************
                    ;       * Other userful constants: *
                    ;       ****************************
                    ;
   0000             b               equ     byte ptr 0
   0000             w               equ     word ptr 0
   0000             false           equ     0
```

```
  FFFF                    true          equ     not false
                          ;
                          ;       ***********************
                          ;       * Start of code area: *
                          ;       ***********************
                          ;
                                  CSEG              ; Small model
                                  ORG     0
0000 E96201        0165           jmp     start
                          ;
0003 CDE0                 tikos:  int     224       ; Execute TIKOS function
0005 C3                           ret

0006 50                   conout: push    ax        ; Output AL to screen
0007 06                           push    es
0008 56                           push    si
0009 B102                         mov     cl,2
000B 8AD0                         mov     dl,al
000D E8F3FF        0003           call    tikos
0010 5E                           pop     si
0011 07                           pop     es
0012 58                           pop     ax
0013 C3                           ret

                          outtext:                  ; Output text constant
0014 AC                           lods    al
0015 0AC0                         or      al,al
0017 7405          001E           jz      outex
0019 E8EAFF        0006           call    conout
001C EBF6          0014           jmps    outtext
001E C3                   outex:  ret
                          ;
                          ;
                          computehxdigit:           ; compute ASCII HEX digit from AL
001F 240F                         and     al,0fh
0021 0430                         add     al,'0'
```

```
CP/M ASM86 1.1  SOURCE: INTV.A86                                    PAGE   3

0023 3C39                        cmp     al,'9'
0025 7602          0029          jbe     chdl
0027 0407                        add     al,7
0029 C3                 chdl:    ret
                        ;
                        computehxbyte:              ; compute ASCII HEX digits
002A 51                          push    cx         ; from AL. Result in AX
002B 53                          push    bx
002C 8AD8                        mov     bl,al    ; save LSB
002E B104                        mov     cl,4
0030 D2E8                        shr     al,cl
0032 E8EAFF        001F          call    computehxdigit
0035 8AE0                        mov     ah,al
0037 8AC3                        mov     al,bl
0039 E8E3FF        001F          call    computehxdigit
003C 5B                          pop     bx
003D 59                          pop     cx
003E C3                          ret
                        ;
                        displayhexbyte:             ; display binary byte as 2 HEX
003F 50                          push    ax
0040 53                          push    bx
0041 E8E6FF        002A          call    computehxbyte
0044 50                          push    ax
0045 8AC4                        mov     al,ah
0047 E8BCFF        0006          call    conout
004A 58                          pop     ax
004B E8B8FF        0006          call    conout
004E 5B                          pop     bx
004F 58                          pop     ax
0050 C3                          ret
                        ;
                        displayhexword:             ; display binary word (AX) as 4
0051 50                          push    ax
0052 8AC4                        mov     al,ah
0054 E8E8FF        003F          call    displayhexbyte
```

```
CP/M ASM86 1.1  SOURCE: INTV.A86                                PAGE    4


0057 58                          pop     ax
0058 E8E4FF         003F         call    displayhexbyte
005B C3                          ret
                    ;
                    outddig:                        ; Output decimal digit
005C 50                     push    ax
005D 0AC0                   or      al,al
005F 7504           0065    jnz     outdd1
0061 8AC2                   mov     al,dl
0063 EB04           0069    jmps    outdd2
0065 0430           outdd1: add     al,'0'
0067 B230                   mov     dl,'0'
0069 52             outdd2: push    dx
006A E899FF         0006    call    conout
006D 5A                     pop     dx
006E 58                     pop     ax
006F C3                     ret

0070 50             decout: push    ax      ; Output decimal number
0071 53                     push    bx
0072 52                     push    dx
0073 B220                   mov     dl,' '
0075 B364                   mov     bl,100
0077 B400                   mov     ah,0
0079 F6F3                   div     bl
007B E8DEFF         005C    call    outddig
007E 8AC4                   mov     al,ah
0080 B30A                   mov     bl,10
0082 B400                   mov     ah,0
0084 F6F3                   div     bl
0086 E8D3FF         005C    call    outddig
0089 8AC4                   mov     al,ah
008B B230                   mov     dl,'0'
008D E8CCFF         005C    call    outddig
0090 5A                     pop     dx
0091 5B                     pop     bx
```

```
0092 58                         pop     ax
0093 C3                         ret

                      test_if_default_vector:          ; Check if standard vector
0094 2BC0                       sub     ax,ax
0096 8EC0                       mov     es,ax
0098 8BF9                       mov     di,cx
009A D1E7                       shl     di,1
009C D1E7                       shl     di,1
009E 268B05                     mov     ax,es:ÆdiÅ
00A1 3B06AA03                   cmp     ax,default_offset
00A5 7508          00AF         jnz     tidex
00A7 268B4502                   mov     ax,es:2ÆdiÅ
00AB 3B06AC03                   cmp     ax,default_segment
00AF C3               tidex:    ret

                      displaymodule:                   ; Display module where
                                                       ; vector points
00B0 51                         push    cx
00B1 BE8D03                     mov     si,offset spactx
00B4 E85DFF        0014         call    outtext
00B7 59                         pop     cx
00B8 2BC0                       sub     ax,ax
00BA 8EC0                       mov     es,ax
00BC 8BF1                       mov     si,cx
00BE 03F6                       add     si,si
00C0 03F6                       add     si,si
00C2 268B4402                   mov     ax,es:2ÆsiÅ      ; Get segment
00C6 BEAC01                     mov     si,offset questx
00C9 3B06AE03                   cmp     ax,tikosseg
00CD 750D          00DC         jnz     dimo
00CF BEC201                     mov     si,offset tikostx
00D2 803EB003FF                 cmp     realtikos,true
00D7 7403          00DC         jz      dimo
00D9 BECD01                     mov     si,offset ccpmtx
00DC 51               dimo:     push    cx
```

```
00DD E834FF     0014           call    outtext
00E0 59                        pop     cx
00E1 C3                        ret
                         ;
                         displaytype:                     ; Display ASCII vector type
00E2 51                        push    cx
00E3 BE9803                    mov     si,offset typseptx
00E6 E82BFF     0014           call    outtext
00E9 59                        pop     cx
00EA BE0001                    mov     si,offset typetab
00ED BA0E00                    mov     dx,typetLG
00F0 3A0C                ditl:  cmp     cl,ÆsiÅ
00F2 7505       00F9           jnz     ditl
00F4 46                          inc   si
00F5 8B34                        mov   si,ÆsiÅ
00F7 EB09       0102             jmps  ditx
00F9 83C603              ditl:  add     si,3
00FC 4A                        dec     dx
00FD 75F1       00F0           jnz     ditl
00FF BEB701                    mov     si,offset ques2tx
0102 51                  ditx:  push    cx
0103 E80EFF     0014           call    outtext
0106 59                        pop     cx
0107 C3                        ret

                         displayvector:                   ; Display vector
0108 E889FF     0094           call    test_if_default_vector
010B 743E       014B           jz      dexit
010D 51                        push    cx
010E BE7F01                    mov     si,offset intno_tx
0111 E800FF     0014           call    outtext
0114 58                        pop     ax
0115 50                        push    ax
0116 E857FF     0070           call    decout
0119 BE9701                    mov     si,offset value_tx
011C E8F5FE     0014           call    outtext
```

```
011F 2BC0                      sub     ax,ax
0121 8EC0                      mov     es,ax
0123 5F                        pop     di
0124 57                        push    di
0125 D1E7                      shl     di,1
0127 D1E7                      shl     di,1
0129 268B4502                  mov     ax,es:2ÆdiÅ
012D 26FF35                    push    es:wÆdiÅ
0130 E81EFF      0051          call    displayhexword
0133 B03A                      mov     al,':'
0135 E8CEFE      0006          call    conout
0138 58                        pop     ax
0139 E815FF      0051          call    displayhexword
013C 59                        pop     cx
013D 51                        push    cx
013E E86FFF      00B0          call    displaymodule
0141 E89EFF      00E2          call    displaytype
0144 BEA003                    mov     si,offset crlf_tx
0147 E8CAFE      0014          call    outtext
014A 59                        pop     cx
014B C3                dexit:  ret

                       get_default_vector:
014C BF9600                    mov     di,150          ; Never-used int function
014F D1E7                      shl     di,1            ; (So far, so good)
0151 D1E7                      shl     di,1
0153 2BC0                      sub     ax,ax
0155 8EC0                      mov     es,ax
0157 268B05                    mov     ax,es:ÆdiÅ
015A A3AA03                    mov     default_offset,ax
015D 268B4502                  mov     ax,es:2ÆdiÅ
0161 A3AC03                    mov     default_segment,ax
0164 C3                        ret
                       ;
                       ;       *****************
                       ;       * Main program: *
```

```
                       ;       *****************
                       ;
0165 1E                start:  push    ds              ; Local stack
0166 17                        pop     ss
0167 BC7904                    mov     sp,offset stack
016A BE2F01                    mov     si,offset signon
016D E8A4FE      0014          call    outtext

0170 E8D9FF      014C          call    get_default_vector
0173 C606B003FF                mov     realtikos,true
0178 B10C                      mov     cl,12           ; Get version no.
017A E886FE      0003          call    tikos
017D 8C06AE03                  mov     tikosseg,es     ; Save segm. of TIKOS
0181 3C22                      cmp     al,22h          ; Ver. 2.2 means TIKOS
0183 740E        0193          jz      tikosthere
0185 C606B00300                  mov   realtikos,false
018A B19A                        mov   cl,9ah          ; Must be CCPM
018C E874FE      0003            call  tikos
018F 8C06AE03                    mov   tikosseg,es
                       tikosthere:
0193 B90000                    mov     cx,0
0196 E86FFF      0108  mloop:  call    displayvector
0199 41                        inc     cx
019A 81F90001                  cmp     cx,256
019E 72F6        0196          jb      mloop
                                                       ; Done, exit program
01A0 B100                      mov     cl,0
01A2 B200                      mov     dl,0
01A4 E95CFE      0003          jmp     tikos

                       ;       *****************
                       ;       * Data segment: *
                       ;       *****************

                               DSEG
                               org     100h
```

```
                       ;
0100                   typetab rb   0         ; Table of special INT
0100 00                        db   0         ; functions
0101 D801                      dw   divZEROtx
0103 01                        db   1
0104 EB01                      dw   singstepTX
0106 02                        db   2
0107 0702                      dw   nmitx
0109 03                        db   3
010A 2A02                      dw   breakptx
010C 04                        db   4
010D 4702                      dw   ovfltx
010F 10                        db   16
0110 6E03                      dw   ibmtx
0112 A0                        db   160
0113 6002                      dw   z80intx
0115 A1                        db   161
0116 7A02                      dw   z80outtx
0118 A2                        db   162
0119 9602                      dw   exectx
011B A3                        db   163
011C B002                      dw   z80segtx
011E A4                        db   164
011F D402                      dw   movbtx
0121 A5                        db   165
0122 FD02                      dw   movwtx
0124 E0                        db   224
0125 2703                      dw   tikoenttx
0127 E1                        db   225
0128 4703                      dw   debenttx

  000E                 typetLG equ  (offset $ - offset typetab)/3
                       ;
                       ;            ******************
                       ;            * User messages: *
                       ;            ******************
```

```
012A 54494B4928                          db 'TIKI('       ; Tiki text standard
012F 0D0A             signon             db cr,lf
0131 566973203830                        db 'Vis 8088 avbrudds-tabell for TIKI-100 '
     383820617662
     72756464732D
     746162656C6C
     20666F722054
     494B492D3130
     3020
0157 313620626974                        db '16 bit prosessor - ver 1.00',cr,lf,lf
     2070726F7365
     73736F72202D
     207665722031
     2E30300D0A0A
0175 000000000000                        db 0,0,0,0,0,0,0,0,0,0
     00000000
017F 417662727564     intno_tx           db 'Avbrudd nr:  ',0,0,0,0,0,0,0,0,0,0,0
     64206E723A20
     200000000000
     000000000000
0197 202020566572     value_tx           db '   Verdi: ',0,0,0,0,0,0,0,0,0,0,0
     64693A200000
     000000000000
     000000
01AC 3F2020202000     questx             db '?',' ',' ',' ',' ',0,0,0,0,0,0
     0000000000
01B7 3F2020202000     ques2tx            db '?',' ',' ',' ',' ',0,0,0,0,0,0
     0000000000
01C2 54494B4F5300     tikostx            db 'TIKOS',0,0,0,0,0,0
     0000000000
01CD 4343504D2000     ccpmtx             db 'CCPM ',0,0,0,0,0,0
     0000000000
01D8 44697669736A     divZEROtx          db 'Divisjon med 0',0,0,0,0,0
     6F6E206D6564
     203000000000
```

```
     00
01EB 50726F736573      singstepTX      db 'Prosessor enkelt steg',0,0,0,0,0,0,0
     736F7220656E
     6B656C742073
     746567000000
     00000000
0207 496B6B65206D      nmitx           db 'Ikke maskerbart avbrudd (NMI)',0,0,0,0,0,0,0
     61736B657262
     617274206176
     627275646420
     284E4D492900
     0000000000
022A 496E6E686F70      breakptx        db 'Innhopp til "debugger"',0,0,0,0,0,0,0
     702074696C20
     226465627567
     676572220000
     0000000000
0247 526567697374      ovfltx          db 'Register overflyt',0,0,0,0,0,0,0,0
     6572206F7665
     72666C797400
     000000000000
     00
0260 4C6573206672      z80intx         db 'Les fra Z80 dataport',0,0,0,0,0,0
     61205A383020
     64617461706F
     727400000000
     0000
027A 536B72697620      z80outtx        db 'Skriv til Z80 dataport',0,0,0,0,0,0
     74696C205A38
     302064617461
     706F72740000
     00000000
0296 5574667C7220      exectx          db 'Utfør Z80 subrutine',0,0,0,0,0,0,0
     5A3830207375
     62727574696E
     650000000000
```

```
     0000
02B0 52657475726E      z80segtx        db 'Returner Z80 segment adresse',0,0,0,0,0,0,0,0
     6572205A3830
     207365676D65
     6E7420616472
     657373650000
     000000000000
02D4 466C79747420      movbtx          db 'Flytt 8-bit ord til/fra Z80 huk.',0,0,0,0,0,0
     382D62697420      ,0,0,0
     6F7264207469
     6C2F66726120
     5A3830206875
     6B2E00000000
     0000000000
02FD 466C79747420      movwtx          db 'Flytt 16-bit ord til/fra Z80 huk.',0,0,0,0,0,
     31362D626974      0,0,0,0
     206F72642074
     696C2F667261
     205A38302068
     756B2E000000
     000000000000
0327 46756E6B736A      tikoenttx       db 'Funksjons-innhopp adresse',0,0,0,0,0,0,0,0
     6F6E732D696E
     6E686F707020
     616472657373
     650000000000
     0000
0347 46756E6B736A      debenttx        db 'Funksjons-innhopp fra "debugger"',0,0,0,0,0,0
     6F6E732D696E      ,0
     6E686F707020
     667261202264
     656275676765
     722200000000
     000000
036E 49424D2D5043      ibmtx           db 'IBM-PC spesial funksjon',0,0,0,0,0,0,0,0
     207370657369
```

```
     616C2066756E
     6B736A6F6E00
     000000000000
     00
038D 202000000000        spactx          db ' ',' ',0,0,0,0,0,0,0,0,0
     0000000000
0398 202D20000000        typseptx        db ' - ',0,0,0,0,0
     0000
03A0 0D0A000000          crlf_tx         db cr,lf,0,0,0
03A5 2954494B49                          db ')TIKI'
                         ;
                         ;               **********************
                         ;               * Scratch variables: *
                         ;               **********************
                         ;
03AA                     default_offset  rw      1       ; default interrupt
03AC                     default_segment rw      1       ; routines
03AE                     tikosseg        rw      1       ; OPSYS segment
03B0                     realtikos       rb      1       ; True if tikos

03B1                     stck            rw      100     ; Local stack
0479 0000                stack           dw      0       ; Produce HEX output
                         ;
                         ;********************** End program ************************
                         end

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:  2%
```

VEDLEGG D

ASCII tabell.

Nedenfor følger en oversikt over alle 7-bits tegnkodene som brukes i TIKOS. Kodene er standard ASCII pluss koder for de norske bokstavene Æ,Ø og Å.

Desimal	Hexadesimal	Tegn
000	00	Null
001	01	Kontroll A
002	02	Kontroll B
003	03	Kontroll C
004	04	Kontroll D
005	05	Kontroll E
006	06	Kontroll F, ACK
007	07	Kontroll G, BELL
008	08	Kontroll H, "<-"
009	09	Kontroll I, TAB
010	0A	Kontroll J, LF
011	0B	Kontroll K
012	0C	Kontroll L, FF
013	0D	Kontroll M, CR
014	0E	Kontroll N
015	0F	Kontroll O
016	10	Kontroll P
017	11	Kontroll Q, XON
018	12	Kontroll R
019	13	Kontroll S, XOFF
020	14	Kontroll T
021	15	Kontroll U, NAK
022	16	Kontroll V, SYN
023	17	Kontroll W
024	18	Kontroll X
025	19	Kontroll Y
026	1A	Kontroll Z
027	1B	Kontroll Æ, ESC
028	1C	Kontroll Ø
029	1D	Kontroll Å
030	1E	Kontroll ^
031	1F	Kontroll _
032	20	(Space)
033	21	!
034	22	"
035	23	#
036	24	$
037	25	%
038	26	&
039	27	'

Vedlegg D. ASCII tabell.

Desimal	Hexadesimal	Tegn
040	28	(
041	29	)
042	2A	*
043	2B	+
044	2C	,
045	2D	-
046	2E	.
047	2F	/
048	30	0
049	31	1
050	32	2
051	33	3
052	34	4
053	35	5
054	36	6
055	37	7
056	38	8
057	39	9
058	3A	:
059	3B	;
060	3C	<
061	3D	=
062	3E	>
063	3F	?
064	40	
065	41	A
066	42	B
067	43	C
068	44	D
069	45	E
070	46	F
071	47	G
072	48	H
073	49	I
074	4A	J
075	4B	K
076	4C	L
077	4D	M
078	4E	N
079	4F	O
080	50	P
081	51	Q
082	52	R
083	53	S
084	54	T
085	55	U
086	56	V
087	57	W
088	58	X
089	59	Y
090	5A	Z
091	5B	Æ
092	5C	Ø

Desimal	Hexadesimal	Tegn
093	5D	Å
094	5E	^
095	5F	_
096	60	`
097	61	a
098	62	b
099	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	æ
124	7C	ø
125	7D	å
126	7E	|
127	7D	DEL

VEDLEGG E

Listen nedenfor refererer kjente bøker som kan være nyttige å ha lest når en skal arbeide med Tiki-100 datamaskinen:

Forfatter	Tittel	Forlag
Rodnay Zaks	CP/M Handbook with MP/M	Cybex
Rodnay Zaks	Programming the Z80	Cybex
Allan R. Miller	Mastering CP/M	Cybex
Andy Johnson-Laird	The Programmer's CP/M handbook	Osborne/ McGraw-Hill

VEDLEGG F

Anmerkninger.

Denne siden er tenkt reservert for brukerens personlige nedtegninger. Kommentarer forøvrig kan sendes Tiki-Data A.S.

ARCTIC RETRO 2023

Tiki-Data a/s
Sinsenveien 53,
0585 Oslo 5

TIKI DATA

Sats: Jonny Bunæs Grafisk A/S
Trykk: Ski Papirindustri A/S